



## **Privatezza e Protezione dei Dati**

Federico Piscitelli

Matricola 970949

federico.piscitelli@studenti.unimi.it

A.A. 2020/2021

Ultimo aggiornamento 7 gennaio 2021

# Indice

<b>1 Privacy and Data Protection in Emerging Scenario</b>	<b>1</b>
1.1 Privacy in Data Publication . . . . .	1
1.1.1 Macrodati vs Microdati . . . . .	2
1.1.2 Protezione per conteggio e frequenza dei macrodati . . . . .	3
1.1.3 Tecniche di protezione per microdati . . . . .	4
1.1.4 Il problema dell'anonimità . . . . .	4
1.1.5 Classificazione degli attributi nelle tabelle di microdati . . . . .	4
1.1.6 Fattori che contribuiscono a rischi nella divulgazione . . . . .	5
1.1.7 Fattori che contribuiscono alla diminuzione dei rischi nella divulgazione . . . . .	5
1.1.8 Misure di rischio . . . . .	5
1.2 K-anonymity . . . . .	6
1.2.1 Generalizzazione e soppressione . . . . .	6
1.2.2 DGH - Domain Generalization Hierarchy . . . . .	6
1.2.3 VGH - Value Generalization Hierarchy . . . . .	7
1.2.4 Tabelle generalizzate tramite la soppressione . . . . .	8
1.2.5 Generalizzazione k-minimale con soppressione . . . . .	9
1.2.6 Criteri soggettivi e oggettivi di scelta . . . . .	10
1.2.7 Granularità di generalizzazione e soppressione . . . . .	10
1.2.8 Esempi di tabelle 2-anonymized . . . . .	11
1.2.9 Algoritmi per calcolare una tabella k-anonima . . . . .	12
1.3 Algoritmi per AG.TS e AG . . . . .	12
1.3.1 k-Optimize algorithm . . . . .	14
1.3.2 Incognito algorithm . . . . .	15
1.3.3 Heuristic algorithm . . . . .	16
1.4 Algoritmi per CS e CG . . . . .	16
1.4.1 Mondrian multidimensional algorithm . . . . .	17
1.4.2 Approximation algorithm . . . . .	18
1.4.3 k-anonymity revisited . . . . .	18
1.5 Attribute disclosure . . . . .	19
1.5.1 l-diversity . . . . .	19
1.5.2 t-closeness . . . . .	19
1.5.3 External knowledge . . . . .	20
1.5.4 Multiple releases . . . . .	20
1.5.5 m-invariance . . . . .	21
1.6 Riassunto delle tecniche viste e altri scenari . . . . .	21
1.7 Applicazioni k-anonymity . . . . .	22
1.7.1 Social Network . . . . .	22
1.7.2 Data mining . . . . .	22
1.7.3 Servizi di localizzazione . . . . .	23
1.7.4 Tipi di privacy . . . . .	23
1.7.5 k-anonymity vs privacy differenziale . . . . .	23
1.8 Altri problemi di Privacy . . . . .	24
1.8.1 Distribuzione di valori sensibili . . . . .	24
1.8.2 Privacy e dati genomici . . . . .	24
1.8.3 Inferenza dal Data Mining . . . . .	24

<b>2 Protezione di Macrodati e Microdati</b>	<b>25</b>
2.1 Macrodata e protezione . . . . .	25
2.2 Tabelle di conteggio . . . . .	26
2.2.1 Sampling . . . . .	26
2.2.2 Regole speciali . . . . .	26
2.2.3 Regole di soglia . . . . .	27
2.3 Tabelle di grandezza . . . . .	28
2.3.1 Soppressione primaria: p-percent . . . . .	29
2.3.2 Soppressione primaria: pq rule . . . . .	30
2.3.3 Soppressione primaria: (n, k) rule . . . . .	31
2.3.4 Soppressione secondaria . . . . .	31
2.4 Microdati e protezione . . . . .	33
2.5 Tecniche di mascheramento . . . . .	33
2.5.1 Sampling . . . . .	34
2.5.2 Local Suppression . . . . .	34
2.5.3 Global Recoding . . . . .	34
2.5.4 Top-coding e Bottom-Coding . . . . .	35
2.5.5 Generalization . . . . .	35
2.5.6 Random noise . . . . .	36
2.5.7 Swapping . . . . .	36
2.5.8 Micro-aggregation (blurring) . . . . .	36
2.6 Tecniche sintetiche . . . . .	36
<b>3 Privacy in Data Outsourcing</b>	<b>38</b>
3.1 Cloud Computing . . . . .	39
3.2 Sfide nella Protezione dei Dati . . . . .	40
3.2.1 Proprietà di Sicurezza . . . . .	40
3.2.2 Requisiti di Accesso . . . . .	41
3.2.3 Architetture . . . . .	41
3.2.4 Combinazioni delle dimensioni . . . . .	41
3.2.5 Problemi da affrontare . . . . .	42
<b>4 Privacy degli utenti</b>	<b>43</b>
4.0.1 User empowerment . . . . .	43
4.0.2 Rilascio diretto . . . . .	43
4.0.3 User privacy preferences . . . . .	44
4.1 Cost-sensitive Trust Negotiation . . . . .	45
4.2 Point-based Trust Management Model . . . . .	45
4.3 Logic-based Minimal Credential Disclosure . . . . .	46
4.4 Privacy Preferences in Credential-based Interactions . . . . .	48
4.4.1 Modellare il Portfolio . . . . .	48
4.4.2 Sensibilità di proprietà e credenziali . . . . .	49
4.4.3 Sensibilità delle associazioni . . . . .	49
4.4.4 Vincoli di rilascio . . . . .	49
4.4.5 Sensibilità del rilascio . . . . .	49
4.4.6 Server Request . . . . .	50
4.4.7 Problema del Rilascio Minimo . . . . .	50
4.4.8 Server side . . . . .	50

<b>5 Privacy and Integrity of Data Storage</b>	<b>51</b>
5.1 Criptazione . . . . .	51
5.1.1 Criptazione e Indici . . . . .	52
5.1.2 Partition-based index . . . . .	54
5.1.3 Hash-based index . . . . .	57
5.1.4 Interval-based queries . . . . .	57
5.2 Searchable encryption . . . . .	58
5.2.1 Order preserving . . . . .	58
5.2.2 Fully Homomorphic Encryption . . . . .	59
5.3 Inference Exposure . . . . .	59
5.3.1 Quotient table . . . . .	60
5.3.2 RCV Graph . . . . .	61
5.4 Bloom filter . . . . .	62
5.5 Integrità dei Dati . . . . .	63
5.6 Selective Encryption and Over-Encryption . . . . .	64
5.7 Frammentazione e criptazione . . . . .	73
5.7.1 Server non comunicanti . . . . .	74
5.7.2 Frammenti non linkabili . . . . .	75
5.8 Frammentazione . . . . .	79
5.8.1 Keep a few . . . . .	79
5.9 Valutazione di Query . . . . .	79
5.9.1 Frammentazione e inferenza . . . . .	83
5.10 Publishing Obfuscated Associations . . . . .	83
5.10.1 Anonymizing Bipartite Graph . . . . .	83
5.10.2 Fragments and Loose Associations . . . . .	85
<b>6 Confidenzialità e integrità delle query</b>	<b>93</b>
6.1 Path ORAM . . . . .	93
6.2 Ring ORAM . . . . .	95
6.3 Shuffle Index . . . . .	96
6.3.1 Rappresentazione a livello logico . . . . .	96
6.3.2 Rappresentazione a livello fisico . . . . .	97
6.3.3 Accesso ai dati . . . . .	97
6.3.4 Cover searches . . . . .	99
6.3.5 Cached searches . . . . .	100
6.3.6 Shuffling . . . . .	100
<b>7 Privacy e dati di locazione</b>	<b>101</b>
7.0.1 Valutazione di LBAC: da $R_{eval}$ a valori di verità . . . . .	103
7.1 Protezione delle informazioni di locazione . . . . .	104

# 1 Privacy and Data Protection in Emerging Scenario

Negli ultimi anni abbiamo assistito ad una continua crescita di:

- database di governi e aziende
- contenuti generati e caricati da utenti, messi a disposizione di fruitori attraverso piattaforme come YouTube, Flickr..
- informazioni identificative di persone, raccolte ogni qual volta una persona si registra ad un sito, effettua la login ad un'applicazione, si registra a una newsletter, partecipa a un sondaggio..

E si sono sempre più diffusi:

- metodi e motivi di distribuzione di dati
  - per effettuare degli studi sui trend o per eseguire inferenze statistiche
  - condividere conoscenze
  - accedere a servizi online
- storage e dispositivi di calcolo esterni, con i vantaggi di:
  - diminuire i costi a dispetto dei servizi offerti
  - grande disponibilità dei servizi e protezione da disservizi di ogni genere

Questo ci porta a voler/dover garantire la privacy e l'intergrità dei dati.

## 1.1 Privacy in Data Publication

Come abbiamo già detto in precedenza, il problema della privatezza nasce quando ho dei dati e voglio condividerli. Solitamente, in passato, i dati venivano usati per scopi statistici e potevamo avere:

- DBMS statistici dove l'utente interroga il DBMS con query statistiche, il DBMS interroga il DB e restituisce la statistica all'utente. Quest'ultimo non può chiedere un dato specifico, ma solo un'aggregazione dei dati (somma, media...)
- Dati statistici dove chi ha il DB produce le statistiche e poi le pubblica.

Possiamo da subito renderci conto che le policy per garantire la sicurezza dei dati devono essere diverse tra questi due sistemi, infatti:

- nel primo caso il controllo deve essere dinamico perchè è l'utente che esegue la query a runtime. Le cose principali da controllare sono:
  - range della query: se il target è troppo piccolo rischio di esporre un singolo individuo, se è troppo grande rischio di esporre piccoli gruppi (perchè posso fare un'intersezione tra gruppo e gruppo più grande per avere come risultato il gruppo più piccolo).

- storico delle query: window history
- collusione: quando due o più parti si mettono insieme per fare query che da soli non avrebbero potuto fare
- nel secondo caso invece è l'owner del db che decide cosa pubblicare, quindi effettua un controllo di tipo statico

### 1.1.1 Macrodati vs Microdati

Possiamo definire dei dati come macrodati quando rilascio statistiche e quando rispondo a query statistiche ossia quando non pubblico i dati veri e propri. I macrodati possono essere identificati in due gruppi (tipi di tabelle):

- tabelle di conteggio/frequenza dove ogni cella contiene il numero di rispondenti o della percentuale di rispondenti

County	Benefit						Total
	\$0-19	\$20-39	\$40-59	\$60-79	\$80-99	\$100+	
A	2	4	18	20	7	1	52
B	-	-	7	9	-	-	16
C	-	6	30	15	4	-	55
D	-	-	2	-	-	-	2

Figura 1: Tabella bidimensionale che mostra il conteggio dei beneficiari diviso per dimensione del benefit e contea

- tabelle di grandezza dove ogni cella della tabella contiene un valore aggregato di una determinata quantità di interesse.

	Hypertension	Obesity	Chest Pain	Short Breath	Tot
M	2	8.5	23.5	3	37
F	3	30.5	0	5	38.5
Tot	5	39	23.5	8	75.5

Figura 2: Media del numero di giorni passati in ospedale dato un disturbo

Quando pubblichiamo dei dati dobbiamo sempre stare attenti a proteggere la privacy di chi sta dietro i dati. Ma esattamente cosa significa proteggere la privacy?

- proteggere l'identità (identity disclosure: un rispondente è identificato dai dati rilasciati)

- proteggere l'informazione sensibile (attribute disclosure: informazioni sensibili riguardanti un rispondente emergono dal rilascio dei dati). Devo stare attento sia quando pubblico dati precisi (es. un numero esatto) sia quando posso fare inferenza (es. valore in un range da 1.3 a 1.7: essendo un range piccolo posso esporre un singolo individuo)
- bloccare la possibilità di fare inferenza sui dati (inferential disclosure: i dati rilasciati rendono possibile la determinazione di caratteristiche riguardanti un rispondente anche se non sono mai stati rilasciati dati su di lui) dove non è la singola informazione ad essere sensibile, ma la possibilità di fare inferenza su di essa la rende sensibile. Ad esempio: il data mining ha lo scopo di trovare correlazioni tra dati, tramite le correlazioni posso fare inferenze e quindi arrivare a dati sensibili.

Nel caso in cui ci sia la identity disclosure un utente terzo è in grado di identificare un rispondente dai dati rilasciati. Abbiamo però una differenza tra macrodati e microdati:

- macrodati: rivelare l'identità non è generalmente un problema fino a quando l'identificazione porta a divulgare informazioni confidenziali (attribute disclosure)
- microdati: l'identificazione è un serio problema siccome i microdati sono dettagliati e l'identity disclosure implica anche l'attribute disclosure

L'inferential disclosure avviene quando le informazioni possono essere ricavate da proprietà statistiche dei dati rilasciati. Questa non sempre rappresenta un rischio:

- dati statistici vengono rilasciati apposta per permettere agli utenti di fare inferenze e capire le relazioni tra le variabili
- le inferenze sono progettate per prevedere comportamenti aggregati, non di individui singoli e sono anche metodi poco precisi per predire dati di singoli

Un primo passo per evitare tutte queste disclosure è sicuramente quello di restringere i dati e restringere l'accesso ai dati: devo limitare i dati che pubblico nelle tabelle (restricted data) e nello stesso tempo devo avere un accesso ristretto, ossia impostare condizioni di accesso ai dati (restricted access). Ad esempio i dati li posso dare solo a certe identità e per certi specifici scopi (purpose of use).

I microdati includono intrinsecamente identificatori esplicativi (nome, indirizzo ip, email) e devono essere tollerati perché sono sensibili e identificanti.

Ulteriori restrizioni potrebbero essere: l'inserimento di un timing entro il quale i dati possono essere tenuti e il divieto di divulgazione ad altri.

### **1.1.2 Protezione per conteggio e frequenza dei macrodati**

Questo tipo di dati include il conteggio di valori o le percentuali di risposte ad un determinato sondaggio.

Per proteggere questi tipi di dati abbiamo diverse tecniche:

- campionamento: pubblicare un sondaggio al posto che un censimento

- regole speciali: specificare delle restrizioni sul livello di dettaglio dei dati
- regole soglia: definisce una determinata cella di una tabella come sensibile se il numero dei rispondenti è minore di un numero specificato

### 1.1.3 Tecniche di protezione per microdati

Le più usate tecniche di protezione dei microdati possono essere classificate come segue:

- masking techniques: trasformo il set originale di dati non rilasciandoli o perturbandoli
  - non perturbative: i dati originali non vengono modificati ma alcuni dati vengono soppressi e altri rimossi (sampling local suppression, generalization)
  - perturbative: i dati originali vengono modificati aggiungendo del rumore (rounding,swapping)
- synthetic data generation techniques: rilascio dati plausibili sintetici, ma non quelli originali
  - fully synthetic: il set dei dati rilasciati contiene solo dati sintetici
  - partially synthetic: il set dei dati rilasciati contiene un mix di dati originali e sintetici

### 1.1.4 Il problema dell'anonimità

I dati vengono de-identificati prima di essere rilasciati, ossia vengono rimossi tutti quei dati identificanti (nome, cognome ...), ma questo non è sufficiente. Questi dati possono essere usati per unire identità con informazioni de-identificate per eseguire un processo di re-identificazione.

### 1.1.5 Classificazione degli attributi nelle tabelle di microdati

- identificatori: identificano univocamente un rispondente (codice fiscale)
- quasi-identificatori: attributi che messi in combinazione con altri possono permettere la reidentificazione
- confidenziali: attributi che contengono informazioni sensibili (malattia di un paziente)
- non confidenziali: attributi che non sono da considerarsi sensibili e il quale rilascio non causa nessun problema di divulgazione

### **1.1.6 Fattori che contribuiscono a rischi nella divulgazione**

Sempre parlando di microdati questi fattori possono generare rischi nella divulgazione:

- esistenza di high visibility records: alcuni record che possono rappresentare caratteristiche uniche come lavori insoliti o stipendi alti
- possibilità di matchare microdata con altre informazioni esterne: esistono individui nella popolazione che possiedono combinazioni di caratteristiche univoche e se queste vengono prese come campione possono generare un disclosure risk.

Oppure posso generare la possibilità di unire dati o aumentare la loro precisione con:

- tanti attributi comuni tra tabelle di microdati e sorgenti esterne
- accuratezza e precisione dei dati (es. data nascita completa o anno di nascita)
- numero e ricchezza delle fonti esterne

### **1.1.7 Fattori che contribuiscono alla diminuzione dei rischi nella divulgazione**

- le tabelle di microdati solitamente contengono un subset della popolazione, quindi può essere che l'utente malevolo non trovi quello che sta cercando
- molte volte i dati recenti non sono disponibili, ma sono accessibili solo quelli più vecchi che creano così un disallineamento temporale
- nelle tabelle di microdati c'è sempre un po'di rumore, dovuto anche a errori
- i dati vengono espressi in formati diversi e quindi è più difficile fare la correzione

### **1.1.8 Misure di rischio**

Per misurare il rischio nella divulgazione dobbiamo considerare:

- la probabilità che il rispondente di cui l'intruso sta cercando informazioni abbia delle informazioni sia nei microdati che in file esterni
- la probabilità che gli stessi valori presenti nei microdati e nei file esterni siano stati salvati in un modo che siano collegabili
- la probabilità che il rispondente cui l'intruso sta cercando informazioni sia unico nella popolazione presente nel file esterno

Se un individuo è unico nella popolazione, allora sarà unico anche in ogni subset. Se invece un individuo è unico in un subset, non per forza sarà unico anche nella popolazione.

## 1.2 K-anonymity

Garantire privacy in senso assoluto non è un'operazione così facile. Di solito ci si limita a garantire un certo livello di anonimia. Per questo entra in gioco k-anonymity che insieme a generalizzazione e soppressione è stato proposto come approccio per proteggere l'identità del rispondente.

K-anonymity ha lo scopo di garantire che i dati rilasciati non possono essere ricollegati a un numero minore di  $k$  persone.

L'idea che sta alla base è quella di non poter utilizzare i quasi identificatori per eseguire operazioni di linking.

Ogni release di dati sarà fatta in modo che ogni combinazione di valori di quasi identificatori può essere collegata ad almeno  $k$  rispondenti. Quindi k-anonymity richiede che ogni valore quasi identificatore che appare nella tabella deve avere almeno  $k$  occorrenze (condizione sufficiente).

Ogni rilascio di dati quindi avrà almeno  $k$  persone che sembrano uguali.

### 1.2.1 Generalizzazione e soppressione

- generalizzazione: il valore di un attributo è sostituito utilizzando valori più generici basato sulla definizione di una generalizzazione gerarchica (al posto che pubblicare tutta la data di nascita, pubblico solo l'anno)
- soppressione: tolgo il dato sensibile (così facendo potrei ridurre il numero di generalizzazioni necessarie a soddisfare il vincolo di k-anonymity)

### 1.2.2 DGH - Domain Generalization Hierarchy

L'idea è quella di costruire per ogni dominio una gerarchia di generalizzazioni che mi dice come andare da un valore più specifico ad uno più generico. Definiamo  $\leq_D$  come una relazione di generalizzazione che definisce un mapping tra il dominio  $D$  e le sue generalizzazioni.

Dati due domini  $D_i$  e  $D_j \in Dom$ ,  $D_i \leq_D D_j$  vuol dire che i valori nel dominio  $D_j$  sono generalizzazioni dei valori in  $D_i$ .

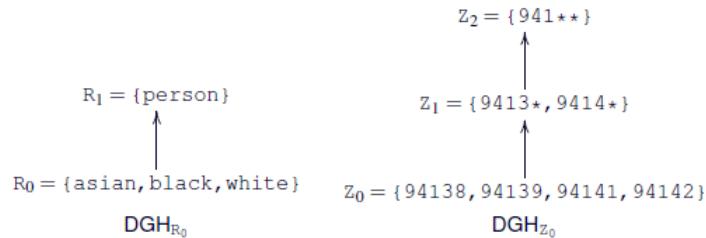
$\leq_D$  implica l'esistenza, per ogni dominio  $D$ , di una  $DGH_D = (Dom, \leq_D)$ :

- $\forall D_i, D_j, D_z \in Dom :$   
 $D_i \leq_D D_j, D_i \leq_D D_z \implies D_j \leq_D D_z \vee D_z \leq_D D_j$   
questa condizione significa ordine totale (catena). I domini sono in relazione a catena.
- L'elemento massimale (la radice della catena) è singleton (ossia ha dentro un solo valore).

L'ordine totale a livello di domini è una catena mentre a livello di tuple di domini è un reticolo (lattice) poiché è il cartesiano delle catene:

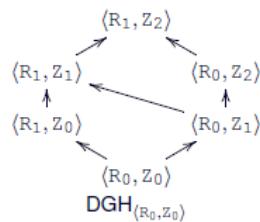
$DT = < D_1, \dots, D_n > | D_i \in Dom \forall i = 1, \dots, n$  la DGH di DT è:  $DGH_{DT} = DGH_{D1} \times \dots \times DGH_{Dn}$

Per esempio una generalizzazione di un singolo attributo (dominio) potrebbe essere:



Dove  $R_1$  e  $Z_2$  sono singleton.

Mentre la generalizzazione di una tupla di domini potrebbe essere:



Come possiamo notare dalla foto abbiamo un reticolo il che comporta le seguenti proprietà:

- ho una relazione d'ordine '
- ho un elemento top e uno bottom
- ho transitività e antisimmetria ossia se una cosa sta sotto a un'altra e l'altra sta sotto di lei, allora stiamo parlando della stessa cosa

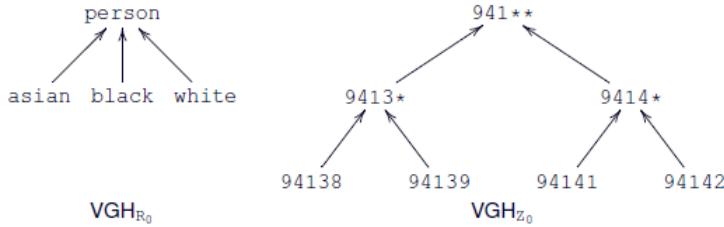
### 1.2.3 VGH - Value Generalization Hierarchy

Definiamo  $\leq_V$  come una relazione di valori di generalizzazione che associa ad ogni valore in un dominio  $D_i$  un valore unico nel dominio  $D_j$ , generalizzazione diretta di  $D_i$ .

$\leq_V$  implica l'esistenza, per ogni dominio  $D$ , di un valore gerarchico di generalizzazione  $VGH_D$ , rappresentato da un albero dove:

- le foglie sono i valori in  $D$
- la radice (ossia il valore più generico) è il valore dell'elemento massimo in  $DGH_D$

Il fatto che sia un albero ci permette di poter dire con certezza che ogni elemento ha un solo padre, cioè c'è un solo cammino che porta un elemento alla radice.



#### 1.2.4 Tabelle generalizzate tramite la soppressione

Siano  $T_i$  e  $T_j$  due tabelle definite dagli stessi attributi. La tabella  $T_j$  è la generalizzazione (tramite soppressione delle tuple) della tabella  $T_i$ . Questo viene denotato come  $T_i \preceq T_j$  se:

1.  $|T_j| \leq |T_i|$  ossia ha un numero minore o uguale di tuple (a livello di cardinalità)
2. il dominio  $dom(A, T_j)$  di ogni attributo A in  $T_j$  è uguale a, o è una generalizzazione di, il dominio  $dom(A, T_i)$  di ogni attributo A in  $T_i$ , ossia per ogni valore in  $T_j$  è una generalizzazione dello stesso valore in  $T_i$
3. è possibile definire una funzione iniettiva che associa ogni tupla  $t_j$  in  $T_j$  tale per cui il valore di ogni attributo  $int_j$  è uguale a, o è una generalizzazione di, il valore del corrispondente attributo in  $t_i$ . Tale funzione non può essere biettiva perché alcune tuple specifiche non "vanno" alla tabella generale perchè non sono state fatte soppressioni

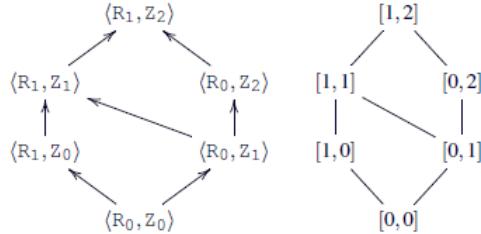
La generalizzazione deve essere eseguita nei giusti termini perchè se è troppo poca rischio di esporre i dati, mentre se è troppa i dati rischiano di non avere più senso.

Race	ZIP	Race	ZIP
asian	94142	person	94141
asian	94141	person	94139
asian	94139	person	94139
asian	94139	person	94139
asian	94139	person	94139
black	94138	person	94139
black	94139	person	94139
white	94139	person	94141
white	94141	person	94141
<hr/> PT		<hr/> GT	

La tabella riportata sopra ha k-anonymity uguale a 2 poichè il gruppo più piccolo è composto da 2 persone (evidenziate in rosso). Le tuple nella tabella GT inoltre non sono associate a tutte quelle della tabella PT (visto che abbiamo soppresso e generalizzato). Possiamo notare anche come la funzione non sia biettiva (biunivoca) infatti non possiamo risalire alla tabella specifica avendo quella generalizzata.

### 1.2.5 Generalizzazione k-minimale con soppressione

Definiamo il Distance vector usando come esempio il reticolo visto in precedenza. Siano  $T_i(A_1, \dots, A_n)$  e  $T_j(A_1, \dots, A_n)$  due tabelle tali che  $T_i \preceq T_j$ . Il distance vector da  $T_j$  a  $T_i$  è il vettore  $DV_{ij} = [d_1, \dots, d_n]$ , dove ogni  $d_z$  con  $z = 1, \dots, n$  è la lunghezza del cammino unico tra  $\text{dom}(A_z, T_i)$  e  $\text{dom}(A_z, T_j)$  nel dominio di generalizzazione gerarchica  $DGH_{Dz}$



Per calcolare la lunghezza del distance vector devo fare la somma, ad esempio: per andare da  $[0,0]$  a  $[1,2]$  ho tre salti poichè ho generalizzato 1 volta R e 2 volte Z.

Siano  $T_i(A_1, \dots, A_n)$  e  $T_j(A_1, \dots, A_n)$  due tabelle tali che  $T_i \preceq T_j$  e sia MaxSup la soglia specifica di soppressione accettata.  $T_j$  è una generalizzazione k-minimale di  $T_i$  se:

1.  $T_j$  soddisfa la k-anonymity facendo la minima soppressione richiesta (minimal suppression):  $T_j$  deve cancellare solo le tuple che servono per soddisfare k-anonymity, cioè non ci sono altre tabelle con la stessa k-anonymity cancellando meno tuple.
2.  $|T_i| - |T_j| \leq \text{MaxSup}$  ossia non sopprimo più di quanto richiesto
3.  $\forall T_z : T_i \preceq T_z$  e  $T_z$  soddisfa le condizioni 1 e 2  $\implies \neg(DV_{i,z} < DV_{i,j})$  cioè non ci deve essere un vettore più specifico che fa lo stesso lavoro.

Per esempio, prendendo il DV della figura riportata in precedenza (ed escludendo ai fini dell'esempio  $[0,0]$  e  $[0,1]$ ), se  $[1,0]$  mi basta allora  $[1,1]$  e  $[1,2]$  non vanno bene.  $[0,2]$  invece è okay perchè sta generalizzando su altro.

Race:R <sub>0</sub>	ZIP:Z <sub>0</sub>	Race:R <sub>1</sub>	ZIP:Z <sub>0</sub>	Race:R <sub>0</sub>	ZIP:Z <sub>1</sub>
asian	94142			asian	9414*
asian	94141	person	94141	asian	9414*
asian	94139	person	94139	asian	9413*
asian	94139	person	94139	asian	9413*
asian	94139	person	94139	asian	9413*
black	94138			black	9413*
black	94139	person	94139	black	9413*
white	94139	person	94139		
white	94141	person	94141		
<hr/> PT		<hr/> GT <sub>[1,0]</sub>		<hr/> GT <sub>[0,1]</sub>	

In questo esempio abbiamo usato  $\text{MaxSup} = 2$  e come possiamo vedere dalla foto sia la seconda che la terza tabella hanno dv uguale a uno rispetto alla prima tabella.

### 1.2.6 Criteri soggettivi e oggettivi di scelta

Possono essere applicati diversi criteri oggettivi di preferenza per scegliere tra due generalizzazioni minimali, come per esempio:

- **minimum absolute distance:** si preferisce la generalizzazione con la distanza assoluta minore, cioè con il numero minore di passi di generalizzazione effettuati. Questo ovviamente non è una soluzione ottimale in alcuni casi, infatti non tiene conto della differenza dei domini: nel nostro caso se generalizziamo la colonna Race le persone diventano tutte uguali, se generalizzo con Zip no
- **minimum relative distance:** non considero ciascun passo con un valore 1, ma lo peso rispetto a dove si colloca nella gerarchia, prendendo quindi quello che relativamente generalizza di meno. Ad esempio: [1,0] fa 1 passo di generalizzazione su una gerarchia alta 1 (cioè  $\frac{1}{1} + \frac{0}{1} = 1$ ) mentre [0,1] fa un passo su una gerarchia alta 2 (cioè  $\frac{0}{1} + \frac{1}{2} = \frac{1}{2}$ ). Sceglio quindi il secondo.
- **maximum distribution:** si preferisce la generalizzazione con il più grande numero di tuple distinte, cioè che lascia una certa diversità tra gruppi di tuple che rispettano comunque la k-anonymity scelta. Nel nostro caso la seconda soluzione dato che crea 3 gruppi invece di due (evidenziati in arancio, azzurro e rosso)
- **minimum suppression:** si preferisce la generalizzazione che sopprime meno tuple, cioè la generalizzazione è quella che mantiene la cardinalità più alta. Nel nostro caso non potremmo adottare questo criterio poiché entrambe le istanze eliminano 2 tuple

Per quanto riguarda i criteri soggettivi invece dobbiamo valutarli noi in base al contesto.

### 1.2.7 Granularità di generalizzazione e soppressione

Entrambi questi metodi possono essere applicati a diversi livelli di granularità:

- la generalizzazione può essere applicata a livello di singola colonna (applico a tutta la colonna la generalizzazione) o singola cella (diverse celle della stessa colonna possono avere diversi livelli di generalizzazione)
- la soppressione può lavorare a livello di riga (rimuovo la tupla), colonna (oscurro i valori di una colonna) e singola cella (una tabella k-anonima può essere ottenuta rimuovendo solo determinate celle)

Il tutto può essere riassunto tramite la seguente tabella:

Generalization	Suppression			
	Tuple	Attribute	Cell	None
Attribute	AG_TS	AG_AS ≡ AG_	AG_CS	AG_ ≡ AG_AS
Cell	CG_TS not applicable	CG_AS not applicable	CG_CS ≡ CG_	CG_ ≡ CG_CS
None	_TS	_AS	_CS	— not interesting

dove AG\_TS significa AttributeGeneralization\_TupleSuppression.

Non applicabile significa che non applico le soluzioni che fanno generalizzazione a granularità fine, ma soppressione a granularità più spessa.

### 1.2.8 Esempi di tabelle 2-anonymized

Race	DOB	Sex	ZIP		Race	DOB	Sex	ZIP	
asian	64/04/12	F	94142		asian	64/04	F	941**	
asian	64/09/13	F	94141		asian	64/04	F	941**	
asian	64/04/15	F	94139		asian	63/03	M	941**	
asian	63/03/13	M	94139		asian	63/03	M	941**	
asian	63/03/18	M	94139		black	64/09	F	941**	
black	64/09/27	F	94138		black	64/09	F	941**	
black	64/09/27	F	94139		white	64/09	F	941**	
white	64/09/27	F	94139		white	64/09	F	941**	
white	64/09/27	F	94141		white	64/09	F	941**	
				PT					AG_TS

La prima soluzione potrebbe essere quella di cancellare l'intera tupla e generalizzare lo ZIP in modo che l'ultima colonna abbia valori uguali

Race	DOB	Sex	ZIP		Race	DOB	Sex	ZIP	
asian	*	F	*		asian	64	F	941**	
asian	*	F	*		asian	64	F	941**	
asian	*	F	*		asian	64	F	941**	
asian	63/03	M	9413*		asian	63	M	941**	
asian	63/03	M	9413*		asian	63	M	941**	
black	64/09	F	9413*		black	64	F	941**	
black	64/09	F	9413*		black	64	F	941**	
white	64/09	F	*		white	64	F	941**	
white	64/09	F	*		white	64	F	941**	
				AG_CS					AG_≡AG_AS

La seconda soluzione nasce dalla generalizzazione della data di nascita e dello ZIP, sopprimendo solo alcuni di questi valori a livello di cella.

La terza soluzione si limita a generalizzare la data di nascita e lo ZIP.

Race	DOB	Sex	ZIP		Race	DOB	Sex	ZIP	
asian	64	F	941**						
asian	64	F	941**						
asian	64	F	941**						
asian	63/03	M	94139						
asian	63/03	M	94139						
black	64/09/27	F	9413*						
black	64/09/27	F	9413*						
white	64/09/27	F	941**						
white	64/09/27	F	941**						
				CG_≡CG_CS					_TS

La quarta soluzione nasce dall'aver effettuato generalizzazioni di diverso livello sulle celle delle colonne DOB e ZIP.

La quinta soluzione potrebbe essere una tuple suppression di tutte le tuple

Race	DOB	Sex	ZIP		Race	DOB	Sex	ZIP
asian	*	F	*		asian	*	F	*
asian	*	F	*		asian	*	F	*
asian	*	F	*		asian	*	F	*
asian	*	M	*		asian	*	M	94139
asian	*	M	*		asian	*	M	94139
black	*	F	*		*	64/09/27	F	*
black	*	F	*		*	64/09/27	F	94139
white	*	F	*		*	64/09/27	F	94139
white	*	F	*		*	64/09/27	F	*

\_AS                    \_CS

La sesta soluzione nasce dalla soppressione completa delle colonne DOB e ZIP.

La settima e ultima soluzione proposta nasce dalla soppressione di celle in quasi tutte le colonne.

Operare a granularità più fine ha come vantaggio l'avere una maggiore utilità in quanto posso lasciare più informazioni nella tabella, ma ha come svantaggi il fatto che:

1. ho una relazione eterogenea: i domini dell'attributo sono eterogenei, cioè diversi, e quindi ho problemi di query
2. il costo computazionale della tabella cresce di molto

### 1.2.9 Algoritmi per calcolare una tabella k-anonima

Il problema di trovare una tabella k-anonima minimale con generalizzazione di attributi e soppressione delle tuple (AG\_TS) è computazionalmente difficile (NP-HARD): ciò che rende il problema complicato è quindi la ricerca della soluzione minimale, cioè sopprimere e generalizzare solo quel che serve, escludendo tutte le altre soluzioni che "ci sono sopra". La k-anonymity è stata pensata alla fine degli anni '90 ma la complessità computazionale è troppo alta (la maggioranza degli algoritmi "esatti" proposti hanno un tempo computazionale esponenziale rispetto al numero degli attributi che compongono i quasi identificatori). Quindi questi algoritmi sono pratici quando il numero QIdegli attributi nell'insieme dei quasi-identificatori è piccolo comparato al numero delle tuple n nella tabella privata PT. Negli anni sono stati proposti numerosi algoritmi per produrre tabelle che rispettino k-anonymity attraverso la generalizzazione e la soppressione di tuple (cioè tramite AG\_TS). Questo perché con AG\_TS possiamo lavorare a livello di schema, perché tengo il problema della ricerca k-minimale nella sua formulazione più semplice.

## 1.3 Algoritmi per AG\_TS e AG

Partiamo con un esempio: so che [0,0] non soddisfa, [1,2] soddisfa. Che calcoli devo fare?

Posso provare tutti i cammini, sia dall'alto che dal basso, e vedere quale combinazione soddisfa. Guardando la gerarchia  $DGH_{DT}$  vediamo che ogni soluzione ha un cammino che posso prendere partendo dalla tabella base oppure dalla tabella più

generale. Se la soluzione corrente non soddisfa i requisiti cambio soluzione, spostandomi in profondità o in ampiezza. Ma in questo modo sto percorrendo tutti i cammini che è computazionalmente oneroso.

Ogni cammino  $DGH_{DT}$  rappresenta una strategia di generalizzazione della tabella PT.

Chiamo **generalizzazione minima locale** il nodo più basso per ogni cammino che soddisfa k-anonymity.

Questo algoritmo porta con se due proprietà:

1. ogni generalizzazione che è minimale globalmente è anche minimale localmente
2. più salgo nella gerarchia, meno tuple devo rimuovere per garantire k-anonymity

Ne deriva che se non hai una soluzione che garantisce k-anonymity e che sopprime meno di MaxSup tuple all'altezza h, allora non puoi avere soluzioni ad altezza più bassa di h (dove per altezza si intende la distanza dalla "radice").

L'algoritmo utilizza una ricerca dicotomica (ricerca binaria) sul reticolo DV:

1. valuta la soluzione ad altezza  $h/2$
2. se esiste una soluzione che soddisfa k-anonymity
  - valuta la soluzione ad altezza  $h/4$
  - altrimenti valuta la soluzione a  $3h/4$
3. fino a che l'algoritmo raggiunge l'altezza minima per la quale c'è un DV che soddisfa k-anonymity

Per ridurre i costi computazionali viene adottata una **matrice distance vector** per evitare la computazione di ogni tabella generalizzata

	Race:R <sub>0</sub>	ZIP:Z <sub>0</sub>	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub> /t <sub>4</sub> /t <sub>5</sub>	t <sub>6</sub>	t <sub>7</sub>	t <sub>8</sub>	t <sub>9</sub>
t <sub>1</sub>	asian	94142	[0,0]	[0,1]	[0,2]	[1,2]	[1,2]	[1,2]	[1,1]
t <sub>2</sub>	asian	94141	[0,1]	[0,0]	[0,2]	[1,2]	[1,2]	[1,2]	[1,0]
t <sub>3</sub>	asian	94139	[1,2]	[1,2]	[1,1]	[0,0]	[0,1]	[1,1]	[1,2]
t <sub>4</sub>	asian	94139	[1,2]	[1,2]	[1,0]	[0,1]	[0,0]	[1,0]	[1,2]
t <sub>5</sub>	asian	94139	[1,2]	[1,2]	[1,0]	[1,1]	[1,0]	[0,0]	[0,2]
t <sub>6</sub>	black	94138	[1,2]	[1,2]	[1,0]	[1,1]	[1,0]	[0,0]	[0,2]
t <sub>7</sub>	black	94139	[1,1]	[1,0]	[1,2]	[1,2]	[1,2]	[0,2]	[0,0]
t <sub>8</sub>	white	94139							
t <sub>9</sub>	white	94141							

Analizziamo la tabella:

- ognuno è distante [0,0] da sé stesso
- t<sub>1</sub> e t<sub>2</sub> diventano uguali con [0,1] cioè con un passo di generalizzazione sullo ZIP
- t<sub>1</sub> e t<sub>6</sub> diventano uguali con [1,2] cioè con una generalizzazione su Race e due su ZIP

Usando questa tabella non devo fare tutti i calcoli poichè posso lavorare seguendo la tabella.

Supponiamo che  $k=2$  e  $\text{MaxSup}=2$

Calcoliamo la prima soluzione ad altezza 1, quindi  $GT_{[1,0]}$  e  $GT_{[0,1]}$

	Race:R <sub>1</sub>	ZIP:Z <sub>0</sub>
t <sub>1</sub>	person	94142
t <sub>2</sub>	person	94141
t <sub>3</sub>	person	94139
t <sub>4</sub>	person	94139
t <sub>5</sub>	person	94139
t <sub>6</sub>	person	94138
t <sub>7</sub>	person	94139
t <sub>8</sub>	person	94139
t <sub>9</sub>	person	94141

	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub> /t <sub>4</sub> /t <sub>5</sub>	t <sub>6</sub>	t <sub>7</sub>	t <sub>8</sub>	t <sub>9</sub>
t <sub>1</sub>	[0,0]	[0,1]	[0,2]	[1,2]	[1,2]	[1,2]	[1,1]
t <sub>2</sub>	[0,1]	[0,0]	[0,2]	[1,2]	[1,2]	[1,2]	[1,0]
t <sub>6</sub>	[1,2]	[1,2]	[1,1]	[0,0]	[0,1]	[1,1]	[1,2]
t <sub>7</sub>	[1,2]	[1,2]	[1,0]	[0,1]	[0,0]	[1,0]	[1,2]
t <sub>8</sub>	[1,2]	[1,2]	[1,0]	[1,1]	[1,0]	[0,0]	[0,2]
t <sub>9</sub>	[1,1]	[1,0]	[1,2]	[1,2]	[1,2]	[0,2]	[0,0]

Soddisfa 2-anonymity sopprimendo t<sub>2</sub> e t<sub>6</sub>

Ora supponiamo  $k=2$  e  $\text{MaxSup}=2$

Calcoliamo la prima soluzione ad altezza 1, quindi  $GT_{[1,0]}$  e  $GT_{[0,1]}$

	Race:R <sub>0</sub>	ZIP:Z <sub>1</sub>
t <sub>1</sub>	asian	9414*
t <sub>2</sub>	asian	9414*
t <sub>3</sub>	asian	9413*
t <sub>4</sub>	asian	9413*
t <sub>5</sub>	asian	9413*
t <sub>6</sub>	black	9413*
t <sub>7</sub>	black	9413*
t <sub>8</sub>	white	9413*
t <sub>9</sub>	white	9414*

	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub> /t <sub>4</sub> /t <sub>5</sub>	t <sub>6</sub>	t <sub>7</sub>	t <sub>8</sub>	t <sub>9</sub>
t <sub>1</sub>	[0,0]	[0,1]	[0,2]	[1,2]	[1,2]	[1,2]	[1,1]
t <sub>2</sub>	[0,1]	[0,0]	[0,2]	[1,2]	[1,2]	[1,2]	[1,0]
t <sub>6</sub>	[1,2]	[1,2]	[1,1]	[0,0]	[0,1]	[1,1]	[1,2]
t <sub>7</sub>	[1,2]	[1,2]	[1,0]	[0,1]	[0,0]	[1,0]	[1,2]
t <sub>8</sub>	[1,2]	[1,2]	[1,0]	[1,1]	[1,0]	[0,0]	[0,2]
t <sub>9</sub>	[1,1]	[1,0]	[1,2]	[1,2]	[1,2]	[0,2]	[0,0]

Soddisfa 2-anonymity sopprimendo t<sub>8</sub> e t<sub>9</sub>

### 1.3.1 k-Optimize algorithm

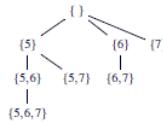
Ordiniamo gli attributi nei QI (quasi identificatori) e i valori nel loro dominio. Successivamente associamo un valore intero ad ogni elemento del dominio, seguendo l'ordine crescente:

Race	ZIP
{[asian] [black] [white]}	{[94138] [94139] [94141] [94142]}
1	4
2	5
3	6
4	7

Una generalizzazione è l'unione di ogni valore degli indici.

L'ordinamento di tali valori ha impatto sul risultato della generalizzazione.

Quindi questo algoritmo produce dei cluster mettendo insieme valori che sono adiacenti. Quando l'algoritmo fa la generalizzazione andrà a mettere insieme i valori adiacenti. Rappresentazione come un insieme: se scrivo (4,6) significa l'insieme (4,5), cioè fino al 6 escluso, e l'insieme dal 6 in avanti (nel nostro esempio 6,7). Poi si mette tutto in un albero:



Possiamo notare come, passando dal nodo padre al nodo figlio vengono messi insieme gruppi che erano separati.

Ogni nodo ha un costo che rappresenta il numero di generalizzazioni e soppressioni dell'anonymizzazione rappresentata dal nodo stesso. Ogni tupla ha un costo associato che riflette la perdita di informazioni associata con la sua generalizzazione o soppressione.

Come facciamo a trovare la soluzione?

k-Optimize visita l'albero utilizzando il metodo depth first, cercando l'anonymizzazione col costo minore. Siccome il numero di nodi in un albero è  $2^{|I|}$  il costo della visita non è molto pratico: adottiamo quindi la tecnica di pruning dell'albero, ossia se il nodo  $n$  non va bene, neanche tutti i suoi figli andranno bene.

Questa determinazione può essere fatta calcolando un margine inferiore sul costo dei nodi nel sottoalbero che ha radice in  $n$ . Se il limite inferiore è maggiore del costo migliore trovato fino ad ora, scarto il nodo  $n$  e i suoi discendenti.

Il problema di questa soluzione è il concetto di adiacenza che non è detto che sia semanticamente rilevante.

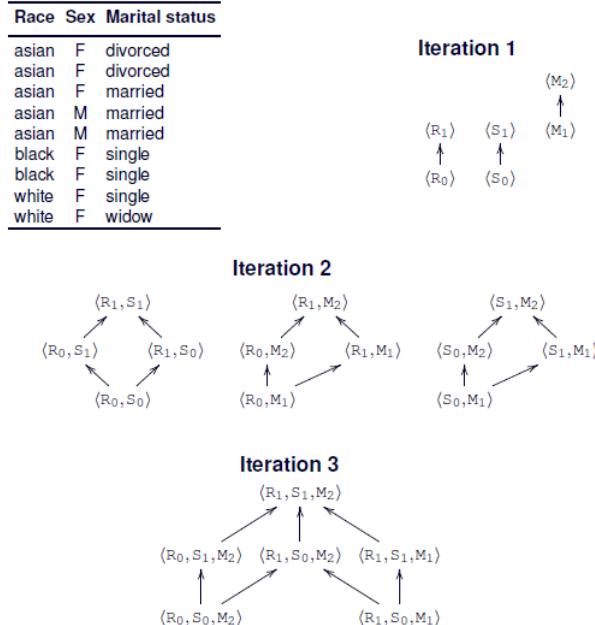
### 1.3.2 Incognito algorithm

Incognito sfrutta questa proprietà: per avere le coppie uguali a due-a-due, anche i singoli valori devono comparire a due-a-due. Esempio: se voglio due occorrenze di Race e ZIP, ci devono essere due valori di R e due valori di Z. Questa è una condizione necessaria ma non sufficiente.

Incognito funziona a iterazioni:

- **Iterazione 1:** controllo che k-anonymity sia rispettata per ogni attributo nei QI, scartando le generalizzazioni che non soddisfano k-anonymity
- **Iterazione 2:** combina le rimanenti generalizzazioni in coppia e controlla k-anonymity per ogni coppia ottenuta.
- ...
- **Iterazione i:** combino le generalizzazioni che soddisfano k-anonymity all'iterazione  $i-1$  e controllo che sia rispettata k-anonymity per ogni  $i$ -upla ottenuta
- **Iterazione  $|QI|$ :** ritorna il risultato finale

Incognito usa un approccio bottom up per la visita di DGHS.



Incognito taglia lo spazio di soluzioni producendo una gerarchia di soluzioni semplificata, ma poi devo andare a cercare il minimo nella gerarchia (ad esempio con la binary search).

### 1.3.3 Heuristic algorithm

Gli algoritmi esatti hanno complessità esponenziale in proporzione a QI. Gli algoritmi euristici proposti sono:

- basati su algoritmi genetici che risolvono il problema di k-anonymity usando un metodo di ricerca stocastico incompleto
- basati su simulated annealing per trovare soluzioni minime locali, ma richiedono un alto tempo di computazione e non garantiscono la qualità della soluzione
- algoritmi euristici top-down per ottenere una tabella k-anonima. Partono dalla soluzione più generale e ad ogni passo tendono a specificare alcuni valori fino a quando non viene violata k-anonymity

Con gli algoritmi euristici non posso specificare margini o bontà della soluzione che andrò a trovare, ma possiamo utilizzare i risultati per calcolare la qualità della soluzione trovata.

## 1.4 Algoritmi per CS e CG

Parliamo di algoritmi che operano a livello di cella effettuando soppressioni e generalizzazioni. Ovviamente questo è più oneroso.

### 1.4.1 Mondrian multidimensional algorithm

Ogni attributo che è un quasi identificatore QI rappresenta una dimensione.

Ogni tupla in PT rappresenta un punto nello spazio definito da QI.

Le tuple con lo stesso valore di QI sono rappresentate dando un valore multiplo al punto nello spazio.

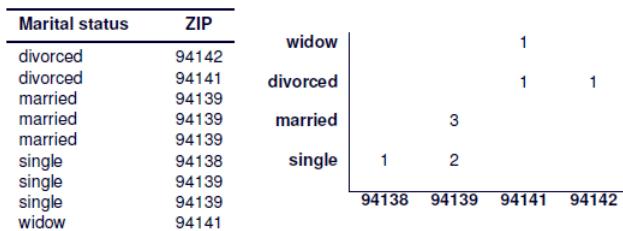
Lo spazio multidimensionale è partizionato dividendo le dimensioni tali che ogni area contenga almeno k occorrenze per valore di punti.

Tutti i punti in una regione sono generalizzati con un unico valore.

Le tuple corrispondenti sono sostituite con quelle generalizzate.

L'algoritmo di Mondrian è flessibile e può operare:

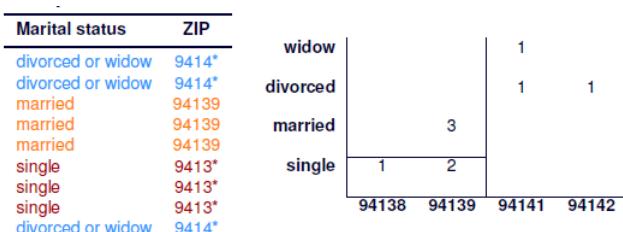
- su diversi attributi
  - singola dimensione
  - dimensione multipla
- con diverse strategie di generalizzazione:
  - generalizzazioni globali
  - generalizzazioni locali
- con diverse strategie di partizionamento
  - partizionamento stretto (non overlapping)
  - partizionamento rilassato (potentially overlapping)
- usando differenti metriche per determinare come dividere ogni dimensione



Come possiamo notare dalla PT i valori vengono ordinati e viene fatto un grafico (numero di assi corrispondente al numero delle colonne) che rappresenta quanti punti ci sono in un determinato incrocio.

Come troviamo la soluzione supponendo di voler soddisfare 3-anonymity?

Devo tagliare lo spazio in modo tale da avere almeno k (nel nostro caso 3) punti per ogni sottospazio.



Il vantaggio di questo algoritmo è che effettua il clustering ottimale senza dover specificare nulla in partenza. Uno svantaggio invece è quello che richiede un ordine totale sul dominio degli attributi e non è sempre possibile farlo.

#### 1.4.2 Approximation algorithm

Non producono la soluzione ottimale ma quella che si avvicina di più a quella che desidero io (ad esempio 1,5-approximation per 2-anonymity)

- algoritmi di approssimazione per CS
  - $O(k \log(k))$ -approximation
  - $O(k)$ -approximation
- algoritmi di approssimazione per CG
  - $O(k)$ -approximation

#### 1.4.3 k-anonymity revisited

Il requisito fondamentale di k-anonymity: per ogni rilascio di una tabella ci deve essere una combinazione di QI che deve almeno matchare k rispondenti.

Quando la generalizzazione è effettuata a livello di attributo questo è uguale a richiedere almeno k occorrenze per ogni valore del QI.

Quando la generalizzazione è effettuata a livello di cella, l'esistenza di almeno k occorrenze è sufficiente ma non è necessaria. In questo caso è sufficiente un requisito meno stringente:

- per ogni sequenza di valori pt in PT[QI] ci sono almeno k tuple in GT[QI] che contiene una sequenza di valori che generalizzano pt
- per ogni sequenza di valori t in GT[QI] ci sono almeno k tuple in PT[QI] che contengono una sequenza di valori per i quali t è una generalizzazione

Race	ZIP	Race	ZIP	Race	ZIP
white	94138	person	9413*	person	9413*
black	94139	person	9413*	person	9413*
asian	94141	asian	9414*	asian	94141
asian	94141	asian	9414*	asian	9414*
asian	94142	asian	9414*	asian	9414*

PT                            2-anonymity                            2-anonymity  
(revisited)

Race	ZIP	Race	ZIP
person	9413*	person	9413*
person	9413*	person	9413*
asian	9414*	asian	94141
asian	9414*	asian	94141
asian	94142	asian	9414*

no 2-anonymity

Con k-anonymity revisited dobbiamo per forza conoscere i dati di partenza per capire se sono protetti.

## 1.5 Attribute disclosure

k-anonymity è vulnerabile ad alcuni tipi di attacco. Prendiamo ad esempio questa tabella

Race	DOB	Sex	ZIP	Disease
asian	64	F	941**	hypertension
asian	64	F	941**	obesity
asian	64	F	941**	chest pain
asian	63	M	941**	obesity
asian	63	M	941**	obesity
black	64	F	941**	short breath
black	64	F	941**	short breath
white	64	F	941**	chest pain
white	64	F	941**	short breath

Gli arancio e i rossi sono esposti perchè i loro record sono uguali. Questo prende il nome di **problema dell'omogeneità**: se nasconde identità in un gruppo omogeneo dove tutti hanno la stessa informazione sensibile, allora non l'ho nascosta. Un altro problema che posso incontrare è quello della **background/external knowledge**, che si basa su una conoscenza pregressa derivante anche da fonti esterne. Ad esempio: un avversario sa che Hellen è una donna bianca e sa che i suoi dati sono in questa tabella. Sa anche che corre due ore al giorno quindi non può avere un problema di fiato corto (quindi da una incertezza di due passo a una incertezza di uno).

### 1.5.1 l-diversity

Un q-block (un insieme di tuple con gli stessi valori per i QI) in T è l-diverso se contiene almeno l "ben-rappresentati" valori diversi per gli attributi sensibili in T. Con l-diversity un avversario deve eliminare almeno l-1 possibili valori per eseguire delle inferenze e capire che un rispondente rispecchia determinati valori.

Prendendo la tabella di prima: il gruppo blu è 3-diverso, quello verde è 2-diverso, quelli arancio e rosso sono 1-diversi.

Una tabella T si dice l-diversa se tutti i q-blocks sono l-diversi (l'attacco basato sull'omogeneità non è più possibile e l'attacco basato su background knowledge diventa più difficile).

Anche l-diversity non risulta essere abbastanza, infatti è vulnerabile a diversi attacchi:

- **skewness attack**: si verifica quando la distribuzione di q-block è differente (ha una distribuzione peculiare) rispetto alla distribuzione della popolazione
- **similarity attack**: si verifica quando i valori sono diversi ma semanticamente simili

### 1.5.2 t-closeness

Un q-block rispetta t-closeness se la distanza tra la distribuzione dei valori degli attributi sensibili nel q-block e nella popolazione considerata è meno di t.

T rispetta t-closeness se tutti i q-block rispettano t-closeness.

$t$ -closeness è monotona con il rispetto della gerarchia di generalizzazione considerata per lo scopo di  $k$ -anonymity.

$t$ -closeness non è facile perché dipende dai nostri dati, può essere che per rispettare  $t$ -closeness i miei dati diventino inutilizzabili perché sono troppo generali.

### 1.5.3 External knowledge

La considerazione delle conoscenze pregresse che possono avere gli eventuali avversari è necessaria quando si ragiona sulla pubblicazione dei dati. Le conoscenze esterne possono essere utilizzate per effettuare:

- **inferenze positive:** osservatore può scoprire che un rispondente ha un certo valore o ha uno di una manciata di valori. Questo caso è considerato il problema principale.
- **inferenze negative:** osservatore può scoprire che un rispondente non ha un certo valore. Generalmente non è considerato un problema ma ovviamente dipende dai dati. Solitamente viene usata come pezzo del puzzle per arrivare all'identificazione/inferenza.

Le informazioni che un attaccante può avere sono diverse e possono arrivare da altrettanti fonti diverse:

- conoscenza rispetto alla persona (esempio: la persona corre due ore al giorno quindi non può avere fiato corto)
- conoscenza rispetto agli altri (esempio: se so che una persona non ha fiato corto, allora l'altra è esposta.)
- conoscenza rispetto agli altri correlati (esempio: DNA simile tra due persone)

Vediamo un esempio:

Name	DOB	Sex	ZIP	Disease		DOB	Sex	ZIP	Disease
Alice	74/04/12	F	94142	aids		74	*	941**	aids
Bob	74/04/13	M	94141	flu	⇒	74	*	941**	flu
Carol	74/09/15	F	94139	flu		74	*	941**	flu
David	74/03/13	M	94139	aids		74	*	941**	aids
Elen	64/03/18	F	94139	flu		64	*	941**	flu
Frank	64/09/27	M	94138	short breath		64	*	941**	short breath
George	64/09/27	M	94139	flu		64	*	941**	flu
Harry	64/09/27	M	94139	aids		64	*	941**	aids

Original table                                  4-anonymized table

La tabella rilasciata è 4-anonima, ma l'avversario sa che Harry che è nato nel 64 e vive nell'area 94139 è nella tabella. Quindi Harry appartiene al gruppo arancio. Da un altro dataset l'avversario sa che George (nato nel 64, vive nell'area 941\*\*) è nella tabella e ha l'influenza. Sempre da sue conoscenze personali l'avversario sa che Harry non ha il fiato corto, quindi l'attaccante può affermare con una confidenza del 50% che Harry ha l'aids.

### 1.5.4 Multiple releases

I dati possono essere soggetti a variazioni frequenti e devono essere pubblicati di costante. Questi rilasci multipli possono causare dei leak delle informazioni siccome un utente malizioso potrebbe correlare questi dataset rilasciati.

$T_1$				$T_2$			
DOB	Sex	ZIP	Disease	DOB	Sex	ZIP	Disease
74	*	941**	aids	[70-80]	F	9414*	hypertension
74	*	941**	flu	[70-80]	F	9414*	gastritis
74	*	941**	flu	[70-80]	F	9414*	aids
74	*	941**	aids	[70-80]	F	9414*	gastritis
64	*	941**	flu	[60-70]	M	9413*	flu
64	*	941**	short breath	[60-70]	M	9413*	aids
64	*	941**	flu	[60-70]	M	9413*	flu
64	*	941**	aids	[60-70]	M	9413*	gastritis

4-anonymized table at time  $t_1$       4-anonymized table at time  $t_2$

Come nel caso precedente, anche qui posso effettuare delle inferenze a partire anche da conoscenze pregresse e ritrovare la malattia di uno specifico paziente.  
Come posso proteggermi da questo attacco?

### 1.5.5 m-invariance

Una sequenza di tabelle di microdati soddisfa m-invariance sse:

- ogni classe di equivalenza include almeno m tuple
- nessun dato sensibile appare più di una volta in ogni classe di equivalenza
- per ogni tupla t, le classi di equivalenza di questa tupla t sono caratterizzate dallo stesso set di dati sensibili

La correlazione di tuple nella sequenza di tabelle non permette a un attaccante di associare meno di m differenti dati sensibili per ogni rispondente

## 1.6 Riassunto delle tecniche viste e altri scenari

- k-anonymity: difendiamo l'identità dei rispondenti confondendoli in gruppi di almeno k elementi
- l-diversity: i gruppi devono avere informazione sensibile diversa altrimenti siamo soggetti ad attacchi di omogeneità
- t-closeness: i gruppi devono avere informazioni sensibili diverse (semanticamente) e con distribuzione il più possibile simile a quella della popolazione reale

Questi algoritmi sono basati su assunzioni che li rendono non sempre applicabili in scenari specifici:

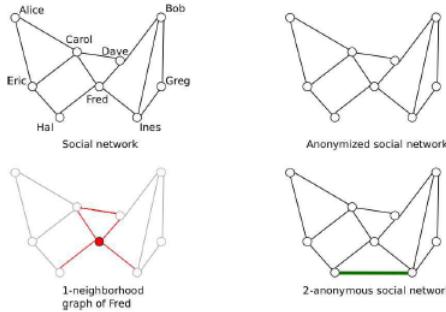
- più tuple che si riferiscono ad uno stesso individuo (ad esempio quando vado all'ospedale e faccio più esami): tecniche  $k^m - anonymity$
- rilascio di tabelle multiple caratterizzate da dipendenze: tecniche multiR k-anonymity
- quasi identificatori multipli: tecniche butterfly
- non ho quasi identificatori predefiniti, infatti definire questi identificatori non è facile: tecniche  $k^m - anonymity$
- rilascio di data stream: tecniche k-anonymity data stream

## 1.7 Applicazioni k-anonymity

k-anonymity può essere applicata ad altri contesti oltre che al rilascio di tabelle di microdati.

### 1.7.1 Social Network

I social network sono soggetti al neighborhood attack dove, data la versione de identificata  $G'$  di un grafo  $G$ , si riesce a ridurre l'incertezza e a volte reidentificare chi sta dietro i singoli nodi.



L'idea per proteggersi da questo attacco è quella di adattare k-anonymity:

- un vertice  $u$  è  $k$ -anonimo se esistono almeno  $k-1$  altri vertici  $v_1, \dots, v$  tali che i sottografi tra i vicini di  $u$  e i vicini di  $v_1, \dots, v$  siano isomorfi
- $G'$  è  $k$ -anonimo se ogni vertice in  $G'$  è  $k$ -anonimo
- se  $G'$  è  $k$ -anonimo, ogni vertice in  $G$  non può essere reidentificato con una confidenza più grande di  $1/k$

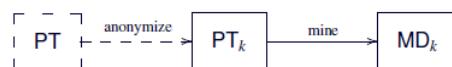
### 1.7.2 Data mining

Privacy riguardo al data mining dipende dalla definizione di privacy a partire da quelle che sono le informazioni sensibili nei dati originali. k-anonymity in data mining ha l'obiettivo di garantire che il risultato del mining non violi k-anonymity. Possono essere principalmente due i tipi di attacco nel mining:

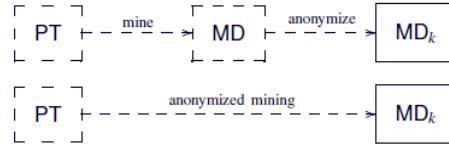
- regole di associazione: faccio mining sui dati per scoprire delle correlazioni nei dati
- regole di classificazione: stabilisco delle regole di classificazione che mi servono per classificare i nuovi dati.

Per proteggere la privacy dei rispondenti posso adottare due tecniche:

- Anonymize and Mine: anonimizzo e poi rilascio la tabella dove chiunque può fare mining



- Mine and Anonymize: faccio mining e poi anonimizzo



### 1.7.3 Servizi di localizzazione

Posso utilizzare anche la posizione come dato per carpire informazioni sensibili e in questo caso k-anonymity entra in gioco in modi diversi:

- se la localizzazione espone l'identità: allargo l'area fino ad avere almeno  $k-1$  individui
- se il luogo è sensibile: offusco l'area per diminuire la precisione
- se la traiettoria della persona è sensibile: creo incertezza nel percorso, offuscando un pezzo

### 1.7.4 Tipi di privacy

**Sintattica:** la definizione di privacy deve soddisfare un certo grado espresso con un numero, ciascun rilascio di dati non può essere riferito a un certo numero ristretto di individui che mi renda comfortable rispetto alla privacy che ho

**Semantica:** le definizioni di privacy sono basate sulla soddisfazione di un requisito semantico di privacy scelto in base al meccanismo del rilascio dei dati, quelli che analizzano i dati vedono un risultato che è indipendente dalla presenza o meno di qualcuno.

**Differenziale:** ha lo scopo di prevenire la capacità di rilevare la presenza o l'assenza di un individuo in un determinato dataset da parte degli avversari. La probabilità di distribuzione dei risultati (cioè il risultato dell'analisi) è essenzialmente la stessa indipendentemente dalla presenza di un individuo. Questa è possibile applicarla essenzialmente a due scenari:

- interattivo: vengono valutate a runtime le query
- non interattivo: vengono rilasciate tabelle di macrodati dove sono già stati fatti dei calcoli

E'posta tendenzialmente aggiungendo rumore casuale ai dati, ma così facendo la veridicità dei dati non è preservata.

### 1.7.5 k-anonymity vs privacy differenziale

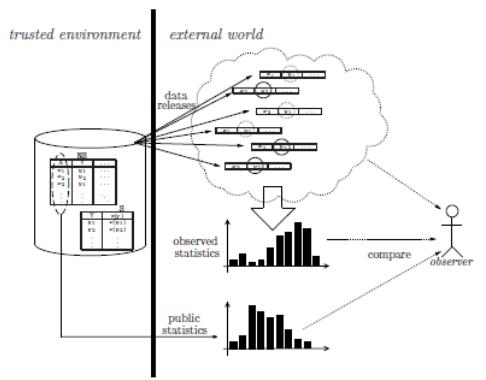
Ognuna delle due ha vantaggi e svantaggi:

	Vantaggi	Svantaggi
k-anonymity	protegge l'info sensibile	non fornisce protezione completa
differential privacy	garantisce più protezione	implementazione più difficile

## 1.8 Altri problemi di Privacy

### 1.8.1 Distribuzione di valori sensibili

Le singole tuple non sono sensibili, ma l'unione di più tuple potrebbe generare un leak di informazioni sensibili (che nel mio dataset non erano rappresentati).  
Esempio: dati medici dei militari; il dato medico del singolo militare non è sensibile. Ma se io guardo l'età posso inferire il tipo di locazione: se sono giovani è un campo di addestramento, se sono vecchi è un quartier generale. Il tipo di locazione è sensibile.



In questo caso si genera un canale di inferenza tra età e posizione. Per contrastare questo canale dobbiamo capire qual è l'informazione sensibile e quando forzare il controllo di esposizione per limitare falsi positivi.

### 1.8.2 Privacy e dati genomici

I dati genomici sono una grande opportunità per la medicina ma a livello di privacy creano non pochi problemi perché:

- identificano il proprietario
- contengono informazioni su ereditarietà etnica, predisposizione a malattie ...
- espongono non solo il proprietario, ma anche i parenti e i discendenti

### 1.8.3 Inferenza dal Data Mining

Esempio reale: una catena di supermercati aveva capito che una ragazza fosse incinta in base agli acquisti. La profilazione ci dice molto di più di quello che c'è nei dati.

Inferenze dai social networks: non è solo quello che io dico, ma anche quello che dicono quelli vicini a me.

## 2 Protezione di Macrodati e Microdati

Con il rilascio dei dati statistici c'è il rischio di inferenza se:

- posso combinare i dati con altre sorgenti
- posso combinare i dati con background ed external knowledge

Abbiamo già trattato la differenza tra DBMS statistici e dati statistici, ma facciamo un breve riassunto:

- i DBMS statistici rispondono solo a query statistiche e per questo bisogna fare controlli a run time per limitare lo spazio di esecuzione della query
- dati statistici dove i dati pubblicati sono solo statistiche e i controlli quindi devono essere fatti su i dati pubblicati

Quindi in un DBMS statistico non posso fare query mirate ma possiamo solo chiedere dati aggregati. Questo significa che non possiamo rispondere a tutte le query.

Name	Sex	Major	Class	Income
Allen	Female	CS	1980	68k
Baker	Female	EE	1980	50k
Cook	Male	EE	1978	70k
Davis	Female	CS	1978	80k
Evans	Male	EE	1981	60k
Frank	Male	CS	1978	76k
Good	Male	CS	1981	64k
Hall	Male	EE	1978	60k
Iles	Male	CS	1979	70k

Query 1: somma degli income degli individui con major EE (il risultato non espone nessun individuo)

Query 2: somma degli income dei maschi con major EE (il risultato non espone nessun individuo)

Query 3: somma degli income delle femmine con major EE (il risultato espone Baker)

La combinazione di queste query è sensibile.

### 2.1 Macrodata e protezione

Come abbiamo visto in precedenza le tabelle di macrodati possono essere classificate come:

- di conteggio/frequenza: ogni cella contiene il numero dei rispondenti che hanno una certa caratteristica
- di grandezza: ogni cella contiene un valore aggregato di una quantità di interesse.

Vediamo nel dettaglio ora qualche tecnica di protezione di questi tipi di tabelle.

## 2.2 Tabelle di conteggio

Tendenzialmente contengono dati provenienti da sondaggi. Le tecniche di protezione adottate includono:

- sampling
- regole speciali
- regole soglia

### 2.2.1 Sampling

Conduciamo un sondaggio e ne pubblichiamo i risultati.

Le stime sono fatte moltiplicando la singola risposta con un sampling weight prima di aggredarli.

Se i pesi non sono pubblicati, questo processo rende meno identificabili gli individui. Le stime devono raggiungere un'accuratezza specifica e i dati che non rispecchiano questi requisiti non vengono pubblicati.

### 2.2.2 Regole speciali

Quando vengono definite tabelle di macrodati su tutta la popolazione devono essere effettuate delle limitazioni nelle divulgazioni. Le **regole speciali** definiscono restrizioni sul livello di dettaglio che può fornire una determinata tabella. Queste regole differiscono in base a organizzazioni e tipi di tabelle.

Esempio: Le regole di SSA (Social Security Administration) proibiscono la pubblicazione di tabelle dove il valore di una cella:

- è uguale a un totale marginale
- permette all'utente di determinare
  - l'età di un individuo all'interno di un intervallo di 5 anni
  - guadagni all'interno di 1000\$ di intervallo
  - benefit all'interno di 50\$ di intervallo

Consideriamo la tabella che segue con questa regola speciale: l'income non deve stare all'interno di un intervallo di 5k

Dept	Income						Total
	[0-21)	(21-23)	(23-25)	(25-27)	(27-29)	29+	
Dept <sub>1</sub>	2	4	18	20	7	1	52
Dept <sub>2</sub>	-	-	7	9	-	-	16
Dept <sub>3</sub>	-	6	30	15	4	-	55
Dept <sub>4</sub>	-	-	2	-	-	-	2

Questa tabella non potrebbe essere rilasciata per diversi motivi:

- il valore di una cella è uguale al totale ( $Dept_4$ )
- il valore di ( $Dept_2$ ) ha dei rispondenti che stanno dentro ad un intervallo di income di 5K

Per ovviare a questo problema potrei ristrutturare la tabella e combinare righe e/o colonne, rendendola pubblicabile.

### 2.2.3 Regole di soglia

Una cella è sensibile se il numero di rispondenti è minore di un certo valore specificato. Quando abbiamo una cella sensibile questa non andrebbe mai rilasciata. Possiamo adottare diverse tecniche per proteggere celle sensibili:

- ristrutturare la tabella e combinare le categorie
- soppressione di celle
- arrotondamento (casuale o controllato)
- modifica dei dati

Prendiamo come esempio la tabella che segue per spiegare le varie tecniche:

Company	Education level				Total
	Low	Medium	High	Very High	
Alfa	15	1	3	1	20
Beta	20	10	10	15	55
Gamma	3	10	10	2	25
Delta	12	14	7	2	35
Total	50	35	30	20	135

Questa tabella ha diversi valori sensibili (con numero di rispondenti < 5).

**Soppressione di celle** È una delle tecniche più utilizzate, ma sopprimere le celle sensibili non è sufficiente (soppressione primaria) poichè il valore nella cella sensibile può essere calcolato dal totale marginale. Per rendere veramente sicuri i dati sensibili dobbiamo applicare una soppressione complementare dove viene soppressa almeno un'altra cella per ogni riga o colonna dove è stata già applicata la soppressione primaria. (Anche con la soppressione complementare è difficile garantire una protezione adeguata).

Per effettuare la soppressione complementare vengono usate tecniche lineari per selezionare quale cella dobbiamo sopprimere e successivamente vengono applicate tecniche di audit per valutare la bontà della soppressione applicata.

Il rischio della soppressione è quello di arrivare ad ottenere una tabella che ha più valori soppressi che valori pubblicati.

Company	Education level				Total
	Low	Medium	High	Very High	
Alfa	15	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	20
Beta	20	10	10	15	55
Gamma	D <sub>4</sub>	D <sub>5</sub>	10	D <sub>6</sub>	25
Delta	D <sub>7</sub>	14	D <sub>8</sub>	D <sub>9</sub>	35
Total	50	35	30	20	135

**Arrotondamento** Per ridurre la perdita di dati dovuta alla soppressione possiamo introdurre un'approssimazione a un multiplo della soglia di sensibilità:

- casuale: decisioni casuali sull'arrotondamento per eccesso o difetto

Company	Low	Medium	High	Very High	Total
Alfa	15	10	10	0	20
Beta	20	10	10	15	55
Gamma	5	10	10	0	25
Delta	15	15	10	0	35
<b>Total</b>	<b>50</b>	<b>35</b>	<b>30</b>	<b>20</b>	<b>135</b>

- controllato: mi assicuro che le somme dei record siano uguali a quelle dei risultati totali marginali

Company	Low	Medium	High	Very High	Total
Alfa	15	10	5	0	20
Beta	20	10	10	15	55
Gamma	5	10	10	0	25
Delta	10	15	5	5	35
<b>Total</b>	<b>50</b>	<b>35</b>	<b>30</b>	<b>20</b>	<b>135</b>

Anche in questo caso vengono utilizzati dei metodi di programmazione lineare per identificare l'approssimazione controllata di una tabella.

Questo metodo ha due svantaggi:

- richiede l'utilizzo di programmi particolari
- non esiste sempre una soluzione per l'approssimazione controllata

**Modifica dei dati** Si modificano i dati alla base (microdati) e poi si calcolano le statistiche con i dati modificati (macrodati). Per applicare questa tecnica ovviamente devo conoscere lo scopo per cui questi dati vengono utilizzati per non "sporcare" eccessivamente e sballare le statistiche.

Questo è un metodo sviluppato dall' U.S. Census Bureau.

Il procedimento da loro applicato è:

1. prendo un campione di record dai microdati
2. trovo una corrispondenza per questi record in qualche altra area geografica, matchando su una serie di specificati attributi importanti
3. swappo tutti gli attributi dei record matchati

### 2.3 Tabelle di grandezza

Soltanamente sono tabelle che contengono la somma (non negativa) di qualcosa (ad esempio il fatturato di determinate aziende). Il nostro obiettivo è limitare le tecniche che si focalizzano sulla valutazione di stime precise dei valori.

Primary suppression rules determina quando un acella potrebbe rivelare l'individuo rispondete a quell'informazione. Tale cella viene considerata sensibile e non può essere rilasciata. Le suppression rules più comuni sono:

- p-percent rule
- pq rule
- (n, k) rule

Queste rules vengono utilizzate per identificare le celle sensibili, verificando dove non è abbastanza difficile determinare una stima precisa di un valore di un rispondente.

### 2.3.1 Soppressione primaria: p-percent

La cella è esposta se si riesce a calcolare il valore del rispondente in maniera troppo accurata. Cosa significa “troppo accurata”? Se l’intervallo di stima (upper-lower) è più vicino al valore esatto rispetto a una percentuale  $p$ .

Più formalmente, una cella è protetta se:

$$\sum_{i=c+2}^N x_i \geq \frac{p}{100} x_1$$

dove

$x_1, \dots, x_N$ : sono i valori dei rispondenti in ordine decrescente

$c$ : dimensione della coalizione di rispondenti ”interpellati” cercando di stimare  $x_1$

$x_1$ : valore più esposto

Esempio: consideriamo i rispondenti che contribuiscono all’entrata totale della città che è uguale a 250k

- Alice: 100k
- Bob: 80k
- Carol: 30k
- David: 20k
- Eve: 10k
- Frank: 3k

Il valore più sensibile è quello di Alice poichè è il più facile da stimare. Quindi se proteggiamo il valore di Alice in modo che non possa essere stimato facilmente, anche tutti gli altri valori saranno protetti.

Qual è la coalizione di  $c = 3$  rispondenti che possono stimare meglio l’entrata di Alice?

Bob, Carol, David, il quale ingresso è 130k, possono stimare che l’ingresso di Alice è tra gli 80K e i 120K (siccome deve essere maggiore o uguale degli 80k di Bob e minore uguale di 120k dato che  $250 - 130 = 120$ ).

La cella è sensibile per ogni  $p \geq 20$

Formalmente la cella è protetta per  $p$  con:

$$\sum_{i=c+2}^N x_i \geq \frac{p}{100} x_1$$

$$\sum_{i=3+2}^N x_i \geq \frac{p}{100} Alice$$

$$\sum_{i=5}^N x_i \geq \frac{p}{100} 100$$

$$\begin{aligned}
Cell - \sum_{i=c+2}^N x_i &\geq p \\
Cell - (Alice + Bob + Carol + David) &\geq p \\
250 - (100 + 80 + 30 + 20) &\geq p \\
20 &\geq p
\end{aligned}$$

### 2.3.2 Soppressione primaria: pq rule

Nel p-percent si assume che non ci siano conoscenze pregresse sui valori dei rispondenti e le agenzie non dovrebbero mai fare queste assunzioni.

Nel pq rule si può esprimere quanta conoscenza pregressa c'è assegnando un valore q che rappresenta quanto accuratamente il rispondente può stimare un valore di un altro rispondente prima che i dati vengano pubblicati ( $p < q < 100$ ).

Formalmente una cella è protetta se:

$$\frac{q}{100} \sum_{i=c+2}^N x_i \geq \frac{p}{100} x_1$$

dove

$x_1, \dots, x_N$ : sono i valori dei rispondenti in ordine decrescente

c: dimensione della coalizione di rispondenti "interpellati" cercando di stimare  $x_1$

$x_1$ : valore più esposto

Se q=100 si assume che i rispondenti non sappiano nulla e non ci sia conoscenza a priori, la formula torna ad essere come quella di p-percent

Esempio: assumiamo che l'abilità di stimare il valore degli altri rispondenti sia q = 80%.

Tutti sanno che l'entrata di Alice è tra 20k e 180k

Qual è la coalizione di c = 3 rispondenti che possono stimare meglio l'entrata di Alice?

Bob, Carol, David, possono ridurre l'incertezza e posizionare l'entrata di Alice tra 80K e i 120K (siccome deve essere maggiore o uguale degli 80k di Bob e minore uguale di 120k dateo che  $250 - 130 = 120$ ).

Formalmente la cella è protetta per p con:

$$\begin{aligned}
\frac{q}{100} \sum_{i=c+2}^N x_i &\geq \frac{p}{100} x_1 \\
\frac{80}{100} \sum_{i=3+2}^N x_i &\geq \frac{p}{100} Alice \\
\frac{80}{100} \sum_{i=5}^N x_i &\geq \frac{p}{100} 100 \\
\frac{80}{100} \sum_{i=5}^N x_i &\geq p
\end{aligned}$$

$$\begin{aligned}
\sum_{i=5}^N x_i &\geq \frac{p}{0.80} \\
Cell - \sum_{i=5}^N x_i &\geq \frac{p}{0.80} \\
Cell - (Alice + Bob + Carol + David) &\geq \frac{p}{0.80} \\
250 - (100 + 80 + 30 + 20) &\geq \frac{p}{0.80} \\
20 &\geq \frac{p}{0.80} \\
16 &\geq p
\end{aligned}$$

### 2.3.3 Soppressione primaria: (n, k) rule

Riguarda il numero di rispondenti in una cella: se un numero piccolo (n o meno) di questi corrispondenti contribuisce ad una larga percentuale (k o più) del valore totale della cella, questa cella è considerata sensibile.

Se ci pensiamo è abbastanza intuitiva come regola, infatti: se una cella è dominata dal valore di un rispondente è facile capire che il totale è un valore che sovrasta il suo.

Solitamente si scelgono n = 1 o n = 2.

Esempio: supponiamo che con n = 2 e k = 70, la cella è considerata sensibile. L'entrata di Alice e Bob è il 70% del totale (180k su 250k).

### 2.3.4 Soppressione secondaria

Una volta che viene identificata una cella sensibile abbiamo due opzioni:

- ristrutturare la tabella e collassare celle fino quando non rimangono più celle sensibili
- soppressione delle celle: non pubblichiamo cell sensibili (primary suppression) e rimuoviamo altre celle (complementary suppression)

Amministrativamente si può anche richiedere un consenso scritto dei rispondenti che li autorizzano a pubblicare certi dati.

Nonostante ciò possono essere rilasciate lo stesso delle celle sensibili visto che:

- unioni implicite delle celle soppresse potrebbero essere sensibili
- l'equazione tra righe e colonne che viene rappresentata dalla tabella pubblicata potrebbe essere risolta

**Audit** Per verificare che dopo la soppressione complementare non ci siano più celle sensibili vengono usate tecniche automatiche di audit.

Se vengono pubblicati i totali la somma delle celle soppresse potrebbe essere derivata. Applicando le regole di sensibilità a queste somme ci assicura che non siano sensibili:

- righe e colonne possono essere visti come sistemi di equazioni lineari
- stimare un limite superiore o inferiore per ogni cella soppressa usando la programmazione lineare.
- se i margini sono troppo vicini al valore originale, la cella è sensibile

Tutto ciò è molto semplice per le tabelle piccole, mentre diventa computazionalmente intrattabile per tabelle grandi.

**Perdita di informazioni** La selezione delle celle complementari deve risultare minima in termini di perdita di informazioni. Potremmo tentare di minimizzare:

- la somma dei valori soppressi
- il numero totale di celle soppresse

Esempio:

Employees by sex and department					
Sex	Dept <sub>1</sub>	Dept <sub>2</sub>	Dept <sub>3</sub>	Dept <sub>4</sub>	Total
Female	1	2	2	1	6
Male	3	2	0	2	7
Total	4	4	2	3	13

Monthly income by sex and department					
Sex	Dept <sub>1</sub>	Dept <sub>2</sub>	Dept <sub>3</sub>	Dept <sub>4</sub>	Total
Female	1800	5600	4200	2500	14100
Male	4500	5800	0	5500	15800
Total	6300	11400	4200	8000	29900

Applichiamo a questa tabella (n, k) rule con n = 1 e k = 90, quindi una cella è sensibile se un rispondente contribuisce a più del 90% del totale.

Ovviamente in questo caso sono Female / Dept<sub>1</sub> e Female / Dept<sub>4</sub>

Applicando soppressione primaria i valori di Female / Dept<sub>1</sub> e Female / Dept<sub>4</sub> sarebbero comunque ricavabili. Applicando invece la soppressione complementare otteniamo la tabella come segue, priva di celle sensibili:

Employees by sex and department					
Sex	Dept <sub>1</sub>	Dept <sub>2</sub>	Dept <sub>3</sub>	Dept <sub>4</sub>	Total
Female	1	2	2	1	6
Male	3	2	0	2	7
Total	4	4	2	3	13

Monthly income by sex and department					
Sex	Dept <sub>1</sub>	Dept <sub>2</sub>	Dept <sub>3</sub>	Dept <sub>4</sub>	Total
Female	D <sub>1</sub>	5600	4200	D <sub>2</sub>	14100
Male	D <sub>3</sub>	5800	0	D <sub>4</sub>	15800
Total	6300	11400	4200	8000	29900

## 2.4 Microdati e protezione

Molte situazioni richiedono il rilascio di dati specifici (microdati). Il vantaggio di pubblicare questo tipo di dato è la crescente flessibilità e disponibilità di informazioni.

Per proteggere l'anonimità del rispondente i data holder rimuovono o criptano spesso identificatori esplicativi come nomi, indirizzi e num. di telefono. Ma questa operazione di deidentificazione dei dati non garantisce l'anonimità. Infatti il rilascio di dati quasi identificanti come lo ZIP, possono essere linkati insieme per risalire al rispondente.

Le strategie di protezione dei dati seguono essenzialmente due strategie:

- ridurre il contenuto di informazioni
- cambiare i dati in maniera che l'informazione sia mantenuta il più possibile

Per proteggere i dati dal rischio di divulgazione si dovrebbe seguire la seguente procedura:

- includere solo un campione di tutta la popolazione
- rimuovere gli identificatori
- limitazione dei dettagli geografici
- limitazione del numero di variabili

Anche la locazione geografica è un dato sensibile da proteggere in quanto è sempre più spesso utilizzato nei microdati ed è identificante per il rispondente.

Le tecniche di protezione dei microdati sono basate sulla limitazione del processo di identificazione riducendo la quantità di informazioni rilasciate e possono essere divise in due categorie:

- mascheramento dei dati: copro i dati non rilasciandoli o introducendo rumore
- sintetizzazione dei dati: rilascio dati non veri, ma plausibili

Queste tecniche possono operare su due tipi diversi di microdati:

- continui: numerici e possiamo definire operazioni aritmetiche su di essi
- categorici: possono assumere determinati e limitati valori e non possiamo eseguire operazioni aritmetiche su di essi

## 2.5 Tecniche di mascheramento

I dati originali vengono trasformati per produrre dei nuovi dati che sono validi per analisi statistiche e che mantengono inviolata la confidenzialità dei rispondenti. Queste tecniche di mascheramento si suddividono in:

- non perturbative: i dati originali non sono modificati, ma alcuni vengono soppressi o ne vengono rimossi dei dettagli
- perturbative: i dati originali vengono modificati (introduciamo rumore)

Lista di tecniche:

Non perturbative		
Tecnica	Continui	Categorici
Sampling	sì	sì
Local Suppression	sì	sì
Global Recoding	sì	sì
Top-coding	sì	sì
Bottom-coding	sì	sì
Generalization	sì	sì
Perturbative		
Tecnica	Continui	Categorici
Resampling	sì	no
Lossy Compression	sì	no
Rounding	sì	no
PRAM	no	sì
MASSC	no	sì
Random Noise	sì	sì
Swapping	sì	sì
Rank Swapping	sì	sì
Micro-aggregation	sì	sì

### 2.5.1 Sampling

Per proteggere la tabella pubblico solo un campione dei microdati, riducendo così il rischio di re identificazione. (Al posto che pubblicare 14 tuple, ne pubblico 11)

### 2.5.2 Local Suppression

Questa tecnica sopprime il valore di un attributo (sostituendolo con un valore vuoto) che risulta significante per la re identificazione, limitando anche la possibilità di fare analisi.

### 2.5.3 Global Recoding

Il dominio di un attributo viene partizionato in intervalli disgiunti (di solito della stessa dimensione) identificati da un'etichetta. Per proteggere i dati quindi viene sostituito il valore di quest'attributo con la sua corrispondente etichetta.

SSN	Name	Race	DoB	Sex	ZIP	MarStat	Holidays	Income
		Asian	64/09/27	F	94139	Divorced	13	260
		Asian	64/09/30	F	94139	Divorced	1	170
		Asian	64/04/18	M	94139	Married	40	200
		Asian	64/04/15	M	94139	Married	17	280
		Black	63/03/13	M	94138	Married	2	190
		Black	63/03/18	M	94138	Married	13	185
		Black	64/09/13	F	94141	Married	15	200
		Black	64/09/07	F	94141	Married	60	290
		White	61/05/14	M	94138	Single	17	170
		White	61/05/08	M	94138	Single	10	300
		White	61/09/15	F	94142	Widow	15	200

Global recoding on Income:  
[150-199]: low, [200-289]: medium, [290-310] high

SSN	Name	Race	DoB	Sex	ZIP	MarStat	Holidays	Income
		Asian	64/09/27	F	94139	Divorced	13	med
		Asian	64/09/30	F	94139	Divorced	1	low
		Asian	64/04/18	M	94139	Married	40	med
		Asian	64/04/15	M	94139	Married	17	med
		Black	63/03/13	M	94138	Married	2	low
		Black	63/03/18	M	94138	Married	13	low
		Black	64/09/13	F	94141	Married	15	med
		Black	64/09/07	F	94141	Married	60	high
		White	61/05/14	M	94138	Single	17	low
		White	61/05/08	M	94138	Single	10	high
		White	61/09/15	F	94142	Widow	15	med

#### 2.5.4 Top-coding e Bottom-Coding

Top-coding definisce un limite superiore (top-code) entro in quale un valore deve stare. Se il valore di un attributo sfiora questo limite il suo valore viene rimpiazzato con top-code.

Bottom-coding definisce un limite inferiore (bottom-code) entro in quale un valore deve stare. Se il valore di un attributo sfiora questo limite il suo valore viene rimpiazzato con bottom-code.

SSN	Name	Race	DoB	Sex	ZIP	MarStat	Holidays	Income
		Asian	64/09/27	F	94139	Divorced	13	260
		Asian	64/09/30	F	94139	Divorced	1	170
		Asian	64/04/18	M	94139	Married	40	200
		Asian	64/04/15	M	94139	Married	17	280
		Black	63/03/13	M	94138	Married	2	190
		Black	63/03/18	M	94138	Married	13	185
		Black	64/09/13	F	94141	Married	15	200
		Black	64/09/07	F	94141	Married	60	290
		White	61/05/14	M	94138	Single	17	170
		White	61/05/08	M	94138	Single	10	300
		White	61/09/15	F	94142	Widow	15	200

Top-coding on **Holidays** for values higher than 30

Bottom-coding on **Holidays** for values lower than 10

SSN	Name	Race	DoB	Sex	ZIP	MarStat	Holidays	Income
		Asian	64/09/27	F	94139	Divorced	13	260
		Asian	64/09/30	F	94139	Divorced	<10	170
		Asian	64/04/18	M	94139	Married	>30	200
		Asian	64/04/15	M	94139	Married	17	280
		Black	63/03/13	M	94138	Married	<10	190
		Black	63/03/18	M	94138	Married	13	185
		Black	64/09/13	F	94141	Married	15	200
		Black	64/09/07	F	94141	Married	>30	290
		White	61/05/14	M	94138	Single	17	170
		White	61/05/08	M	94138	Single	10	300
		White	61/09/15	F	94142	Widow	15	200

#### 2.5.5 Generalization

Consiste nel rappresentare un valore di un dato attributo usando dei valori più generici. È basato sulla definizione di una gerarchia di generalizzazione, dove il valore più generico è la radice e le foglie sono i valori più specifici.

Possono essere create diverse tabelle di microdati in base al livello di generalizzazione.

### 2.5.6 Random noise

Perturba i dati sensibili aggiungendo o moltiplicando il valore con una variabile random con una data distribuzione. Pubblicare questa distribuzione aumenta il rischio di divulgazione dei dati.

### 2.5.7 Swapping

Vengono scambiati i valori degli attributi di una piccola parte dei record che mattonano con altri record nella stessa tabella (ossia sono molto simili).

### 2.5.8 Micro-aggregation (blurring)

Consiste nel raggruppare tuple in piccoli aggregati di dimensione k.

Al posto che pubblicare i singoli valori degli individui si pubblica una media del gruppo. Questi gruppi vengono formati utilizzando criteri di massima similità.

SSN	Name	Race	DoB	Sex	ZIP	MarStat	Holidays	Income
	Asian	64/09/27	F	94139	Divorced	13	260	
	Asian	64/09/30	F	94139	Divorced	1	170	
	Asian	64/04/18	M	94139	Married	40	200	
	Asian	64/04/15	M	94139	Married	17	280	
	Black	63/03/13	M	94138	Married	2	190	
	Black	63/03/18	M	94138	Married	13	185	
	Black	64/09/13	F	94141	Married	15	200	
	Black	64/09/07	F	94141	Married	60	290	
	White	61/05/14	M	94138	Single	17	170	
	White	61/05/08	M	94138	Single	10	300	
	White	61/09/15	F	94142	Widow	15	200	

Raggruppiamo le tuple in base a Sex e MarStat. Calcoliamo la media delle entrate per ogni gruppo e la sostituiamo con il valore dei singoli:

SSN	Name	Race	DoB	Sex	ZIP	MarStat	Holidays	Income
	Asian	64/09/27	F	94139	Divorced	13	215	
	Asian	64/09/30	F	94139	Divorced	1	215	
	Asian	64/04/18	M	94139	Married	40	213	
	Asian	64/04/15	M	94139	Married	17	213	
	Black	63/03/13	M	94138	Married	2	213	
	Black	63/03/18	M	94138	Married	13	213	
	Black	64/09/13	F	94141	Married	15	245	
	Black	64/09/07	F	94141	Married	60	245	
	White	61/05/14	M	94138	Single	17	235	
	White	61/05/08	M	94138	Single	10	235	
	White	61/09/15	F	94142	Widow	15	200	

## 2.6 Tecniche sintetiche

Siccome il contenuto statistico dei dati non è correlato alle informazioni date da ogni rispondente possiamo costruire un modello che rappresenta e sostituisce i dati forniti.

La cosa importante quando si costruiscono dati sintetici è che i dati originali e quelli sintetici devono fornire la stessa qualità di analisi statistica.

Il vantaggio principale di questa classe di tecniche è che i dati non sono correlati a nessun rispondente e quindi non può essere effettuata la re identificazione. Lista di tecniche:

Totalmente sintetiche		
Tecnica	Continui	Categorici
Bootstrap	sì	no
Cholesky	sì	no
Decomposition		
Multiple	sì	sì
Imputation		
Maximum Entropy	sì	sì
Latin Hypercube Sampling	sì	sì

Parzialmente sintetiche		
Tecnica	Continui	Categorici
IPSO	sì	no
Hybrid Masking	sì	no
Random Response	no	sì
Blank and Impute	sì	sì
SMIKe	sì	sì
Multiply Imputed Partially Synthetic Dataset	sì	sì

### 3 Privacy in Data Outsourcing

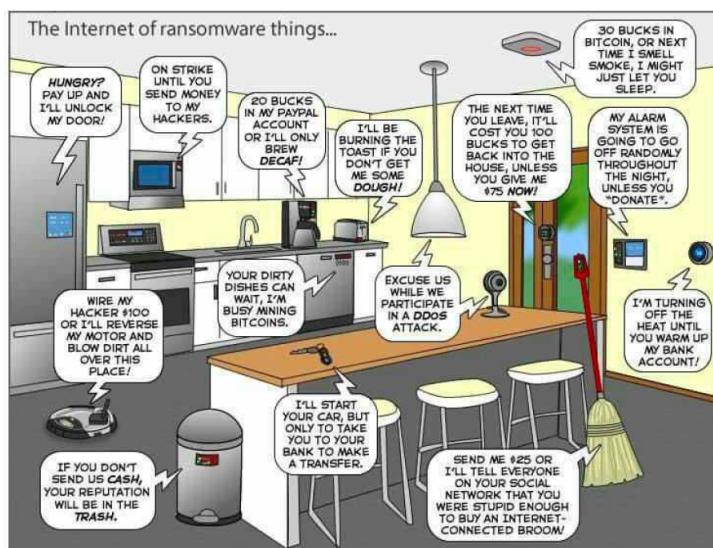
L'avanzamento nell'ICT ha completamente cambiato la nostra società, le infrastrutture e i servizi sono molto più potenti, efficienti e complessi. Questo sviluppo nell'ICT ha aperto le porte per la creazione di una smart society.

Vantaggi degli smart devices:

- migliori meccanismi di protezione
- business continuity e disaster recovery che mi danno la possibilità di continuare ad operare anche quando qualcosa va storto
- migliori capacità di prevedere e rispondere ad eventuali attacchi e intrusioni

D'altro canto però:

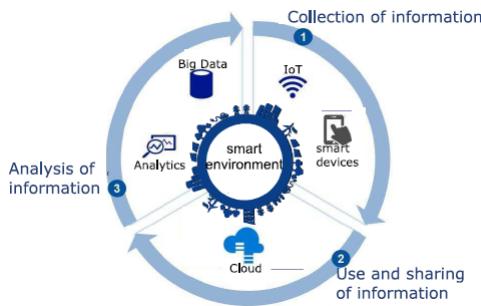
- sistemi molto complessi dove il collegamento debole diventa un facile punto di attacco
- esplosione di danni e violazioni
- perdita del controllo di dati e processi (dove sono i miei dati e chi li gestisce?)



Diventa essenziale quindi chiedersi: cosa dobbiamo proteggere?

- Infrastruttura
- Singole componenti
- Comunicazione
- Dati

Il ruolo dei dati in uno scenario smart è fondamentale:



### 3.1 Cloud Computing

Il cloud permette agli utenti e alle organizzazioni di affidarsi a provider esterni per immagazzinare, processare e accedere ai loro dati, essenzialmente per:

- grandi possibilità di configurazione e scalabilità conveniente
- dati e servizi sempre accessibili
- infrastrutture scalabili per le applicazioni

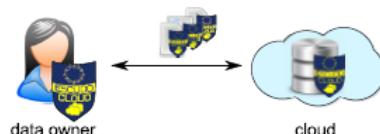
Ma questo avviene tutto a discapito del controllo dei propri dati, introducendo così nuovi problemi di sicurezza e privacy.

I Cloud Service Provider (CSPs) offrono una buona protezione, il problema è che questa protezione è relativa solo al perimetro del cloud provider. All'interno, se il provider ha la chiave può vedere i dati.

Perché al provider dovrebbe servire la chiave? Per dare servizi: se vuoi la risposta a una query, il provider deve poter leggere i dati. Questo implica una totale fiducia nel provider che ha completo accesso ai nostri dati (Google Drive, iCloud)



Ci sono anche soluzioni dove il provider non possiede la chiave e questo rende molto oneroso riuscire fare delle query. Abbiamo quindi una protezione dei dati, ma limitate funzionalità siccome il CSP non può accedere ai dati (Boxcryptor, SpiderOak). Negli ultimi tempi si stanno cercando delle soluzioni ibride dove si ha sia la protezione dei dati (rispetto all'esterno e rispetto al provider) che le funzionalità complete. In questo approccio vengono considerate sicure solo le operazioni dell'utente.



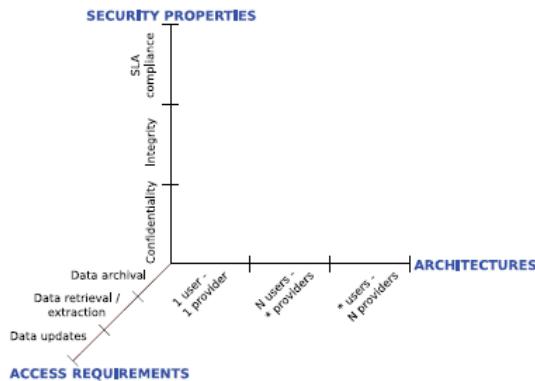
Quando si affronta la protezione dei dati bisogna cercare di **minimizzare** il rilascio e l'esposizione per evitare:

- correlazione tra diverse sorgenti di dati
- inferenza indiretta usando dati esterni

**deidentificare  $\neq$  anonimizzare**

## 3.2 Sfide nella Protezione dei Dati

Queste sfide sono caratterizzate principalmente da tre dimensioni, riassunte nello schema seguente:



Nelle sezioni seguenti andremo ad analizzare una alla volta.

### 3.2.1 Proprietà di Sicurezza

- **Confidenzialità:** mantenere la confidenzialità vuol dire permettere l'accesso e la visione solo a chi è autorizzato e comprendono:
  - dati che memorizzo all'esterno, sia rispetto al mondo esterno, sia rispetto al CSP
  - identità degli utenti che utilizzano il sistema
  - azioni che gli utenti fanno sui dati in quanto in alcuni casi non è confidenziale il dato in sé, ma lo è l'operazione che fa l'utente su di esso
- **Integrità:** più complessa della confidenzialità perché stiamo parlando di persone autorizzate che modificano i dati. Non è possibile sempre garantire l'integrità, possiamo invece fornire una specie di backup che mi garantisce "l'integrità" (capisco se i dati sono stati compromessi e li recupero con la tecnica di ridondanza degli stessi). Questa deve essere garantita:
  - rispetto ai dati che ho memorizzato all'esterno
  - rispetto ai risultati delle computazioni e delle query (più complicato perché effettuato dinamicamente)

- **Conformità al Service Level Agreement (SLA):** garantire conformità rispetto allo SLA che si è sottoscritto, il che comprende:
  - garanzia e certificati
  - disponibilità

### 3.2.2 Requisiti di Accesso

- **Archiviazione dei dati:** semplice archiviazione dei dati
  - upload e download
  - solo protezione dei dati
- **Estrazione e Retrieve dei Dati:**
  - query selettive, dove il provider entra nei dati, li legge e mi fornisce il risultato della query
  - protezione sulla computazione delle query (potrebbero essere confidenziali e il CSP non dovrebbe essere in grado di conoscere)
- **Aggiornamento dei Dati:** dati dinamici che possono cambiare
  - supporto all'accesso e all'aggiornamento dei dati
  - protezione della privacy delle operazioni sui dati

### 3.2.3 Architetture

- **1 user - 1 provider**
  - protezione dei dati in storage
  - query specifiche
  - privacy nelle query e nell'integrità dei dati
- **n users - \* providers:**
  - controllo delle autorizzazioni e degli accessi
  - più scrittori
- **\* users - n providers:**
  - controllo e regolazione dello scambio dei dati tra più providers

### 3.2.4 Combinazioni delle dimensioni

Ogni combinazione delle istanze delle dimensioni identifica nuovi problemi e nuove sfide. Le proprietà di sicurezza che devono essere garantite dipendono dai requisiti di accesso e dall'assunzione di fiducia nel provider coinvolto.

I provider possono essere:

- curiosi
- prigri
- maliziosi

### **3.2.5 Problemi da affrontare**

- privacy degli utenti
- protezione dei dati
- esecuzione delle query
- accessi privati
- controlli di accesso
- integrità e correttezza dei dati
- pubblicazione dei dati
- esecuzione di query collaborative

## 4 Privacy degli utenti

Il desiderio degli utenti è quello di non dover sempre dichiarare la loro identità quando eseguono qualcosa nel cloud. Per questo entrano in gioco:

- tecniche di comunicazione anonima
- privacy basata sulla locazione
- controllo di accesso su attributi
- preferenze di privacy che l'utente ha quando interagisce col sistema

### 4.0.1 User empowerment

Gli utenti potrebbero voler specificare le policy che regolano le informazioni divulgati quando usano servizi esterni per condividere le loro risorse (Facebook) o quando rilasciano informazioni in interazioni digitali (carte di credito quando accedo a un determinato servizio).

Dobbiamo quindi proteggere due aspetti:

- rilascio diretto, regola da chi, quando e con che scopo un utente è d'accordo a rilasciare delle informazioni
- uso secondario dei dati, regola l'uso e la condivisione dei dati che erano usati per uno scopo primario

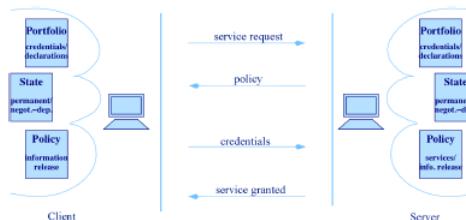
### 4.0.2 Rilascio diretto

La comunità di ricerca è stata molto attiva in questo ambito e ha prodotto diversi approcci che regolano le interazioni tra le parti basate sulla definizione di meccanismo **attribute-based access control**:

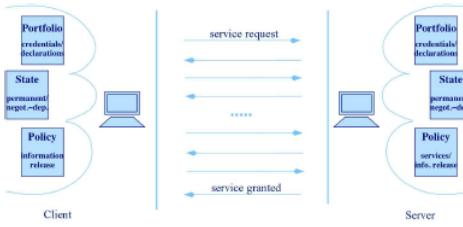
- cosa un utente può fare in base agli attributi che presentano nel loro certificato
- il controllo di accesso non risponde più sì/no, ma risponde con una lista di requisiti che il richiedente del servizio deve soddisfare per ottenere l'accesso

Inoltre non dobbiamo pensare a garantire la sicurezza solo al server ma anche il client vuole le sue garanzie, magari introducendo qualche forma di negoziazione.

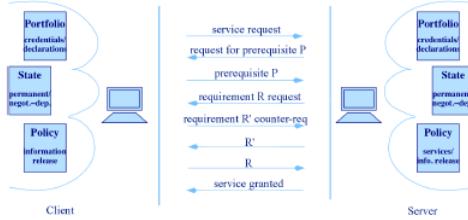
Vediamo tre modalità di Interactive Access Control: Controllo di accesso senza condizioni dal client (troppo semplice):



Controllo di accesso senza condizioni dal client, ma con negoziazione (troppo complicato):



Controllo di accesso senza condizioni dal client, con two step interaction:



Esistono tecnologie che supportano ABAC e sono:

- U-Prove/Idemix: forniscono un avanzato sistema di gestione delle credenziali
- XACML: standard per interoperazioni di controllo di accesso

#### 4.0.3 User privacy preferences

Le specifiche di controllo di accesso non sempre riescono a coprire anche il problema dal lato dell'utente, infatti non permettono all'utente di esprimere la preferenza di rilasciare alcune informazioni rispetto a delle altre.

- **Context-based:** rilascio la mia carta di credito solo quando devo effettivamente pagare
- **Forbidden disclosures:** non voglio rilasciare allo stesso server il mio vero nome e il mio nickname
- **Sensitive associations:** il mio ZIP code e la data di nascita sono più sensibili quando sono insieme
- **Limited disclosure:** ti dico se sono maggiorenne ma non ti dico la mia età precisa
- **History-based:** ti do la provincia di residenza invece del mio telefono se possiedi già il mio indirizzo
- **Proof-based:** posso dimostrare qualcosa senza rilasciare il documento, ad esempio dimostro di essere italiano senza rilasciarti il passaporto
- **Non-linkability:** preferisco rilasciare dei pezzi di informazioni che se messe insieme mi rendono il meno identificabile possibile

Successivamente approfondiremo qualche approccio, tra cui:

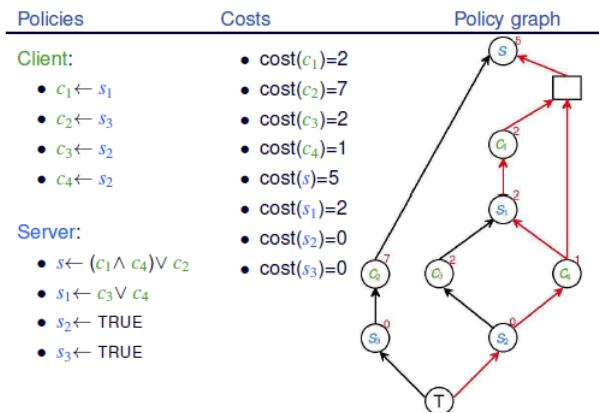
- Cost-sensitive trust negotiation
- Point-based trust management model
- Logic-based minimal credential disclosure
- Privacy preferences in credential-based interactions

#### 4.1 Cost-sensitive Trust Negotiation

Due parti (client e server) interagiscono tra di loro stabilendo fiducia reciproca scambiandosi credenziali (trust negotiation protocol).

Credenziali e policies sono associate a un costo, dove più le credenziali o le policies sono sensibili, più il costo è alto.

L'obiettivo è quello di minimizzare il costo totale di sensibilità di credenziali e policies durante una trust negotiation.



In questo esempio il client ha quattro credenziali:  $c_1, c_2, c_3$  e  $c_4$ .

Il server ha tre credenziali  $s_1, s_2, s_3$  e il servizio  $s$  a cui dà accesso.

Le politiche che sono implementate sono del tipo "se mi dai questo, ti rilascio questo". Quindi, lato server:  $s_2$  e  $s_3$  vengono sempre rilasciati,  $s_1$  viene rilasciato solo se mi presenti  $c_3$  o  $c_4$  e  $s$  viene rilasciato se mi presenti  $c_1$  e  $c_4$  oppure in alternativa  $c_2$ . Il client invece dice: rilascio  $c_1$  se mi dai  $s_2$ ,  $c_2$  se mi dai  $s_3$  e  $c_3$  e  $c_4$  se mi rilasci  $s_2$ .

Il percorso scelto alla fine è evidenziato in rosso e viene scelto proprio quello perché è minima la somma dei costi.

Questo metodo fornisce un meccanismo per regolare il rilascio delle credenziali in accordo con la loro sensibilità e pone il suo focus sulla negoziazione più che sul controllo del client. Inoltre minimizzare il costo totale ha applicazioni limitate. Una combinazione lineare di costi potrebbe non essere sempre desiderata.

#### 4.2 Point-based Trust Management Model

Chi possiede il servizio dà un certo numero di punti per ogni credenziale. Quindi per accedere ad un certo servizio servono un tot di credenziali (questa soglia viene

mantenuta privata). Anche il client valuta le sue credenziali con un punteggio privato che indica la sensibilità dell'informazione. L'obiettivo diventa quindi quello di raggiungere il punteggio del server dando meno informazioni sensibili possibili.

SERVER				
	College ID	Driver's license	Credit card	SSN
Point value	3	6	8	10

CLIENT				
	College ID	Driver's license	Credit card	SSN
Sensitivity score	10	30	50	100

Supponiamo che bisogna fornire almeno 10 punti per accedere a una risorsa.  
Le opzioni che ha il client sono:

- SSN [10 punti, 100 di sensibilità]
- College ID, Credit Card [11 punti, 60 di sensibilità]
- Driver's license, Credit Card [14 punti, 80 di sensibilità]

L'opzione numero due è quindi la migliore.

Il problema è che bisogna continuare a fornire dati fino a che non abbiamo raggiunto la soglia (Credential Selection Problem).

La soluzione è convertire il problema in un knapsack problem e risolverlo con un approccio di programmazione dinamica. Più esattamente viene utilizzato un protocollo di programmazione dinamica sicuro e a due parti:

- il server e il client calcolano insieme la somma ottima dei punti senza rivelare i loro punteggi
- il protocollo usa homomorphic encryption, ossia l'operazione viene fatta sui dati criptati (equivale a decriptare, fare l'operazione e criptare)

Il modello point-based ha quindi queste caratteristiche:

- client e server devono mettersi d'accordo sulle credenziali e avere quindi un'ontologia comune
- ogni credenziale è considerata come un'unità
- mette il focus sulla negoziazione più che sul controllo del client

### 4.3 Logic-based Minimal Credential Disclosure

Le parti sono coinvolte in una negoziazione fidata dove il rilascio delle credenziali è regolato da delle policies. Ogni credenziale è un singolo attributo. Le diverse policies formano un grafo.

Alice's policy	On-line book shop's policy	Negotiation paths
$c_{name} \leftarrow \text{TRUE}$	$\text{purchase} \leftarrow p_{register} \wedge p_{payment}$	
$c_{bdate} \leftarrow c_{bbb}$	$p_{register} \leftarrow (c_{name} \wedge c_{bdate} \wedge (c_{email} \vee c_{pcode})) \vee$	
$c_{telephone} \leftarrow c_{bbb}$	$c_{id} \vee c_{passport} \vee$	
$c_{email} \leftarrow c_{bbb}$	$((c_{name} \vee c_{email}) \wedge c_{id}) \vee$	
$c_{pcode} \leftarrow c_{bbb}$	$(c_{bname} \wedge c_{baccount}) \vee$	
$c_{id} \leftarrow c_{bbb}$	$(c_{credit\_card} \wedge c_{pin})$	
$c_{passport} \leftarrow c_{bbb}$	$c_{bbb} \leftarrow \text{TRUE}$	
$c_{bname} \leftarrow c_{bbb} \wedge c_{osc}$	$c_{osc} \leftarrow \text{TRUE}$	
$c_{baccount} \leftarrow c_{bbb} \wedge c_{osc}$		
$c_{credit\_card} \leftarrow c_{bbb} \wedge c_{osc}$		
$c_{pin} \leftarrow c_{bbb} \wedge c_{osc}$		

La tabella di negoziazione può essere vista come una tabella binaria dove 0 indica il non rilascio, 1 indica il rilascio.

	Name	bdate	telephone	email	pcode	id	Passport	bname	baccount	credit_card	pin
$S_1$	1	1	0	1	0	0	0	1	1	0	0
$S_2$	1	1	0	1	0	0	0	0	0	1	1
$S_3$	1	1	0	0	1	0	0	1	1	0	0
$S_4$	1	1	0	0	1	0	0	0	0	1	1
$S_5$	0	0	0	0	0	1	0	1	1	0	0
$S_6$	0	0	0	0	0	1	0	0	0	1	1
$S_7$	0	0	0	0	0	0	1	1	1	0	0
$S_8$	0	0	0	0	0	0	1	0	0	1	1
$S_9$	1	0	0	0	0	1	0	1	1	0	0
$S_{10}$	1	0	0	0	0	1	0	0	0	1	1
$S_{11}$	0	0	0	1	0	1	0	1	1	0	0
$S_{12}$	0	0	0	1	0	1	0	0	0	1	1

La preferenza di default è ovviamente quella di non rilasciare una credenziale piuttosto che rilasciarla:

$$\implies 0 \succ_i 1$$

con i che rappresenta la i-esima credenziale. I set di rilascio sono comparati attraverso la **Pareto composition** ( $\succ_p$ ), dove:

$s_i$  domina  $s_j$  se  $s_i$  mostra un valore migliore o uguale a  $s_j$  rispettando la preferenza di credenziali ed è strettamente migliore con il rispetto di almeno una credenziale. Ad esempio: presi  $s_5 : [0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0]$  e  $s_9 : [1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0]$  notiamo come  $s_5[1] \succ_1 s_9[1]$  mentre gli altri valori sono uguali. Questo vuol dire che  $s_5$  domina  $s_9$  e più precisamente  $s_5 \succ_p s_9$ .

Introdurre delle gerarchie ci permette di poter definire la preferenza di rilascio di una credenziale rispetto ad un'altra e introduce anche la transitività.

Le caratteristiche di questo modello sono:

- utenti coinvolti nella scelta del set da rilasciare
- si basa sugli attributi e non sulle credenziali
- non sempre è facile esprimere le preferenze degli utenti tra gruppi di attributi

- la possessione di credenziali sensibili non è considerata
- rilasci vietati non sono supportati

## 4.4 Privacy Preferences in Credential-based Interactions

L'obiettivo è quello di permettere all'utente di regolare effettivamente il rilascio delle sue proprietà e delle sue credenziali:

- identificare i requisiti e i concetti che devono essere catturati
- organizzare le proprietà dell'utente e le credenziali nel suo portfolio
- abilitare l'utente ad esprimere quanto ritiene sensibili il rilascio di certe proprietà presenti nel portfolio
- fornire possibili approcci tecnici per supportare le preferenze dell'utente
- fornire una base per investigare approcci user friendly per regolare il rilascio delle proprietà

### 4.4.1 Modellare il Portfolio

Abbiamo citato in questi punti il portfolio dell'utente, ma esattamente cosa vuol dire modellare il suo portfolio?

- Credenziali: certificati verificati e firmati da qualcuno che certifica le proprietà. Una credenziale di solito possiede un type, un identifier e un issuer (chi l'ha rilasciata)
- Dichiarazioni: una credenziale autofirmata

Il portfolio ha diversi tipi di proprietà:

- indipendenti: il valore dipende solo dal proprietario delle credenziali
- dipendenti: il valore dipende dalla credenziale certificante (il numero di carta di credito varia in base alla carta di credito)

Allo stesso modo, il portfolio ha diversi tipi di credenziali:

- atomiche: quando rilascio un certificato lo rilascio tutto (ad esempio la carta d'identità, perchè con lei rilascio il nome, il cognome ecc ecc)
- non atomiche: posso selezionare cosa rilasciare (ad esempio la patente, dove non sono obbligato a dare tutte le informazioni)

Un rilascio è un subset del portfolio dell'utente che soddisfa:

- certificabilità: ogni proprietà è certificata da una credenziale
- atomicità: se viene rilasciata una proprietà di una credenziale atomica, tutte le sue proprietà vengono rilasciate

Parti diverse di un portfolio hanno diversi livelli di sensibilità (l'utente infatti potrebbe preferire rilasciare una proprietà x piuttosto che una proprietà y). Per poter effettuare questa diversificazione introduciamo delle **etichette di sensibilità** tra cui vale l'ordine parziale (ossia non abbiamo maggiore o minore stretto, ma maggiore e minore uguale) e sulle quali è definito un operatore di composizione  $\oplus$  (la composizione di due etichette di sensibilità è un'altra etichetta di sensibilità)

#### 4.4.2 Sensibilità di proprietà e credenziali

$\lambda(A)$ : sensibilità della proprietà A presa individualmente.

$\lambda(c)$ : sensibilità dell'esistenza della credenziale c

#### 4.4.3 Sensibilità delle associazioni

$\lambda(A)$ : sensibilità dell'associazione  $A = \{A_i, \dots, A_j, c_k, \dots, c_n\}$ , il cui rilascio congiunto porta:

- più informazioni che il rilascio di ogni elemento in A (**sensitive view**)
- meno informazioni del rilascio di ogni elemento in A (**dependency**)

#### 4.4.4 Vincoli di rilascio

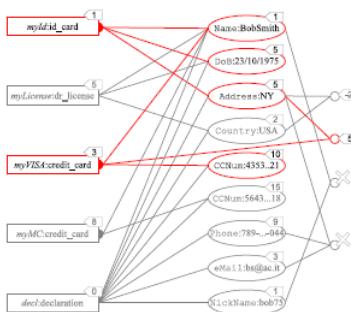
Dato il set  $A = \{A_i, \dots, A_j, c_k, \dots, c_n\}$  di elementi il cui rilascio deve essere controllato abbiamo:

- **forbidden view**: il rilascio di A è proibito
- **disclosure limitation**: almeno n elementi in A possono essere rilasciati

Un rilascio è valido se non viene violato nessun vincolo

#### 4.4.5 Sensibilità del rilascio

La sensibilità  $\lambda(D)$  di un rilascio D è la somma delle etichette di sensibilità che sto rilasciando, considerando proprietà, credenziali e associazioni. Ad esempio:



In questo caso abbiamo che le proprietà valgono  $1+5+5+10$ , le credenziali  $3+1$  e le associazioni 5. In totale  $\lambda(D) = 30$

#### 4.4.6 Server Request

La richiesta R è una semplice disgiunzione (OR) di richieste semplici, dove una richiesta semplice r è data da una congiunzione (AND) di termini.

Ad esempio:

$$R = r_1 \vee r_2$$

$$r_1 = id.\{Name, Address\}$$

$$r_2 = cc.\{Name, CCNum\}$$

Data la richiesta R come fa il client a decidere quali credenziali rilasciare? Devo trovare un insieme che soddisfi la richiesta del server e che abbia la minore sensibilità possibile.

#### 4.4.7 Problema del Rilascio Minimo

Un rilascio D:

- soddisfa R, ossia soddisfa almeno una delle r in R
- soddisfa r se per ogni credenziale richiesta il client presenta una credenziale del tipo richiesto e che certifichi quelle proprietà
- è minima se non esiste un rilascio D' migliore

Calcolare il problema del minimo rilascio è NP-hard e abbiamo due modalità per risolverlo:

- sfruttare la rappresentazione del grafo e vedere se esistono euristiche in grado di risolvere il problema
- tradurlo in un problema di soddisfacibilità booleana così da usare SAT solver: in pratica ho delle variabili booleane e una formula da soddisfare. Devo trovare quindi un assegnamento alle variabili che mi renda la formula vera. Max-SAT è massimizzare il numero delle variabili soddisfatte, ma a noi serve il contrario, dato che dobbiamo minimizzare (ci basta tradurre Max-Sat in forma negativa)

#### 4.4.8 Server side

XACML è uno standard utilizzato per interoperabilità di politiche di controllo di accesso. Questo però ha diversi limiti:

- non fornisce la specifica della proprietà che deve essere certificata dal certificato
- non supporta le astrazioni
- non supporta la politica del dialogo (per comunicare all'utente le policy)
- non supporta condizioni ricorsive (per esprimere policy basate su catene o su delegazioni)

## 5 Privacy and Integrity of Data Storage

La comunità di ricerca è stata molto attiva e ha prodotto diversi contributi e avanzamenti:

- soluzioni per proteggere la confidenzialità dei dati salvati
- indicizzazione che supporta differenti tipi di query
- valutazione di esposizione delle inferenze
- integrità dei dati
- accesso selettivo

Le soluzioni per proteggere i dati possono essere basate su:

- criptazione
- criptazione e frammentazione
- frammentazione

### 5.1 Criptazione

Il server può essere **honest-but-curious** cioè protegge l'integrità ma non è affidabile sulla protezione della sensibilità. La confidenzialità può essere protetta aggiungendo uno strato di criptazione sui dati sensibili (e per ragioni di performance, la criptazione, è fatta a livello di tuple):

- a livello di tabella, ma se dovessi fare una query dovrei decriptare tutto
- a livello di tuple, ma non potrei fare proiezioni
- a livello di attributo, ma non potrei fare selezioni
- a livello di cella, richiede tante operazioni di criptazione e decriptazione

Per ragioni di confidenzialità i CSPs che immagazzinano dati non posso decriptarli per processarli o per accederci. Per questo abbiamo bisogno di un meccanismo che ci permetta di accedere a questi dati che sia effettivo ed efficiente e che non dia la possibilità di fare inferenze.

Esistono diversi approcci:

- **Keyword search:** ricerca di stringhe sui dati criptati, non è una query ma string matching. Il server non deve decriptare.
- **Homomorphic encryption:** si eseguono le operazioni sui dati criptati. Questo è molto costoso.
  - Fully Homomorphic: operazione sul criptato ha lo stesso risultato che l'operazione fatta sui dati in chiaro e poi criptati
  - Addition and Multiplication Homomorphic: rispetto ad addizione e moltiplicazione

- Order Preserving Homomorphic: posso valutare se è maggiore, minore o uguale
- Deterministic Homomorphic: mi dice se ho corrispondenza 1 a 1
- Random Homomorphic: non mi dice nulla
- **Encryption Schemas:** ogni colonna può essere criptata con uno schema diverso a seconda di determinate condizioni che vengono valutate su di esse
- **Onion Encryption:** critto i dati a cipolla dove ogni livello supporta l'esecuzione di specifiche operazioni SQL. Il problema con questo metodo è che quando la "cipolla è pelata", rimane pelata; se il criptato è order preserving allora la crittografia fa leak dell'ordine.
- **Indexes:** vengono allegati dei metadati ai dati e sono utili per fare le query. Ad esempio il WHERE della query non è valutato sul criptato ma sugli indici associati.

### 5.1.1 Criptazione e Indici

Come è fatta una tabella criptata indicizzata?

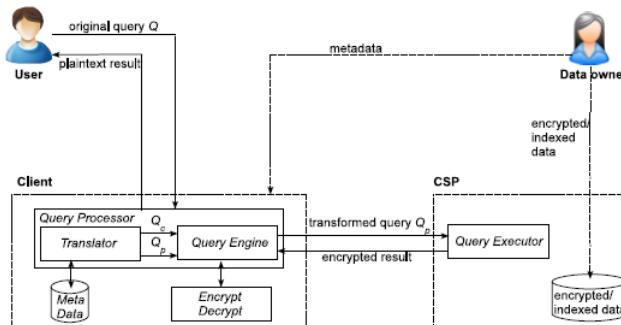
Accounts					
Account	Customer	Balance	$I_A$	$I_C$	$I_B$
Acc1	Alice	100	$\pi$	$\alpha$	$\mu$
Acc2	Alice	200	$\varpi$	$\alpha$	$\kappa$
Acc3	Bob	300	$\xi$	$\beta$	$\eta$
Acc4	Chris	200	$\rho$	$\gamma$	$\kappa$
Acc5	Donna	400	$\varsigma$	$\delta$	$\theta$
Acc6	Elvis	200	$\tau$	$\varepsilon$	$\kappa$

Accounts <sup>k</sup>					
Counter	Etuple	$I_A$	$I_C$	$I_B$	
1	x4Z3fx2ShOSM	$\pi$	$\alpha$	$\mu$	
2	mNHg1oC010p8w	$\varpi$	$\alpha$	$\kappa$	
3	WslaCvfyF1Dxw	$\xi$	$\beta$	$\eta$	
4	Jp08eLTvgwV1E	$\rho$	$\gamma$	$\kappa$	
5	qctG6XnFNDTQc	$\varsigma$	$\delta$	$\theta$	
6	4QbqCeq3hxZhklU	$\tau$	$\varepsilon$	$\kappa$	

La tupla viene criptata interamente dando come risultato la stringa contenuta in Etuple. Per ciascun attributo sul quale devo valutare delle condizioni, specifico dei metadati (nel nostro caso le colonne Account, Customer e Balance diventano gli indici contenuti nelle colonne  $I_A$ ,  $I_C$ ,  $I_B$ ).

Come funziona questo processo?



Il data owner:

1. critta i dati
2. li indicizza
3. li memorizza sul server

L'utente per accedere ai dati deve per forza passare a un Client trusted:

1. fa una query
2. la query viene tradotta in una query che opera su gli indici
3. prende il risultato
4. decripta il risultato
5. fa una post computazione
6. ritorna i risultati all'utente

Come indici possiamo utilizzare diversi valori:

- **Direct (1:1):** uso un valore vero o una codifica del valore vero. In questo caso ad un indice corrisponde un dato specifico. Vantaggi: approccio semplice e preciso per query di uguaglianza (riesco a valutare più precisamente le query). Svantaggi: preserva la distinguibilità dei valori reali e quindi soggetto ad attacchi di inferenza (in particolare attacchi di frequenza). Esempio:

Patients				Patients <sup>k</sup>					
SSN	Name	Illness	Doctor	ID	Etuple	Is	In		
123...89	Alice	Asthma	Angel	1	x423lX2Sh0SM	π	κ	α	δ
234...91	Bob	Asthma	Angel	2	mNHq1oC010p8W	ω	ω	ω	δ
345...12	Carol	Asthma	Bell	3	WslaCvlyh1DxW	ξ	λ	α	ν
456...23	David	Bronchitis	Clark	4	Jp08elTVgwV1E	ρ	ν	β	γ
567...34	Eva	Gastritis	Dan	5	qctG6XnFNDTQc	ι	μ	α	σ
232...11	Eva	Stroke	Ellis	6	kotGBXnFND1aW	χ	σ	β	ψ

Angel è più esposto perchè è l'unico che compare due volte

- **Bucket (n:1):** inizio ad avere collisioni dove valori diversi si mappano allo stesso valore indicizzato. Vantaggi: supporta ancora query di uguaglianza , la collisione rimuove la distinguibilità. Svantaggi: i risultati possono contenere risultati che io non ho richiesto e quindi devo effettuare del post processing per filtrare solo le tuple che mi servono, siamo ancora vulnerabili a attacchi di inferenza (frequenza)

Patients				Patients <sup>k</sup>					
SSN	Name	Illness	Doctor	ID	Etuple	Is	In		
123...89	Alice	Asthma	Angel	1	x423lX2Sh0SM	π	κ	α	δ
234...91	Bob	Asthma	Angel	2	mNHq1oC010p8W	ω	ω	ω	δ
345...12	Carol	Asthma	Bell	3	WslaCvlyh1DxW	ξ	λ	α	ν
456...23	David	Bronchitis	Clark	4	Jp08elTVgwV1E	ρ	ν	β	γ
567...34	Eva	Gastritis	Dan	5	qctG6XnFNDTQc	ι	μ	α	σ
232...11	Eva	Stroke	Ellis	6	kotGBXnFND1aW	χ	σ	β	ψ

Sapendo le reali frequenze posso capire che l'asma non è in beta.

- **Flattened (1:n):** un valore in chiaro può essere mappato a più indici. Vantaggi: riduce la possibilità di ricevere attacchi di inferenza. Svantaggi: rimane vulnerabile all'osservazione dinamica (se faccio le query chiedendo di Eva mi ritorneranno tutti gli indici e so che saranno associati ad Eva)

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
678...11	Eva	Stroke	Ellis

Patients <sup>k</sup>			
Id	Etuple	I <sub>0</sub>	I <sub>N</sub>
1	x4Z3tfX2ShOSM	$\pi$   $\kappa$   $\alpha$   $\delta$	
2	mNHg1oC010p8W	$\omega$   $\omega$   $\alpha$   $\delta$	
3	WslaOvfyf1Dxw	$\xi$   $\lambda$   $\alpha$   $\nu$	
4	JpO8e1VgwV1E	$\rho$   $\nu$   $\beta$   $\gamma$	
5	qctG6Xnf-NDTQe	$\iota$   $\mu$   $\alpha$   $\sigma$	
6	kotG8Xnf-NDtaW	$\chi$   $\sigma$   $\beta$   $\psi$	

Eva era doppia e l'ho mappata con due indici diversi.

### 5.1.2 Partition-based index

Consideriamo un arbitrario attributo col testo in chiaro  $A_i$  in uno schema relazionale R con dominio  $D_i$ .  $D_i$  è partizionato in un numero di sottoinsiemi non sovrapponibili, chiamate partizioni, che contengono valori contigui.

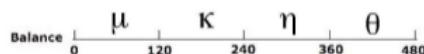
Data una tupla t in r, il valore dell'attributo  $A_i$  per t appartiene ad una partizione: la funzione  $ident_{R,A_i}(p_j)$  assegna ad ogni partizione  $p_j$  di attributi  $A_i$  in R un identificatore.

Il valore che corrisponde all'indice è l'unico valore associato con la partizione a cui l'attributo  $t[A_i]$  appartiene:

$$Map_{R,A_i}(v) = ident_{R,A_i}(p_j)$$

dove  $p_j$  è la partizione che contiene v.

Map può essere random o order-preserving (l'ordine degli indici è uguale a quello sulle tuple in chiaro).



In questo esempio:

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

Questo metodo supporta le query dove le condizioni sono formule booleane nella forma:

- Attribute op Value
- Attribute op Attribute

dove op include  $\{=, <, >, \leq, \geq\}$ . Più nello specifico:

- $A_i = v$ , il mapping è definito come

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

Nell'esempio  $Map_{cond}(Balance = 100) \implies I_B = Map_{Balance}(100) = \mu$

- $A_i < v$ , il mapping dipende dalla funzione  $Map_{A_i}$  se è random o order-preserving:

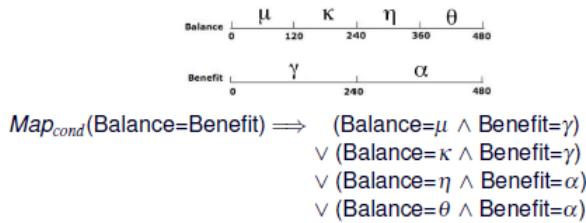
- se è order-preserving:  $Map_{cond}(A_i < v) \implies I_i \leq Map_{A_i}(v)$
- se è random: si controlla se l'attributo appartiene in una delle partizioni che potrebbe contenere il valore cercato:  
 $Map_{cond}(A_i < v) \implies I_i \in Map_{A_i}^<(v)$

- $A_i > v$  è simmetrico rispetto al punto precedente

- $A_i = A_j$  la traduzione è fatta considerando tutte le possibili coppie di partizioni che si sovrappongono. Formalmente

$$Map_{cond}(A_i = A_j) \implies \vee_{\phi} (I_i = ident_{A_i}(p_k) \wedge I_j = ident_{A_j}(p_l))$$

dove  $\phi$  è  $p_k \in \text{partition}(A_i)$ ,  $p_l \in \text{partition}(A_j)$ ,  $p_k \cap p_l \neq \emptyset$



- $A_i < A_j$  il mapping dipende dalle funzioni  $Map_{A_i}$  e  $Map_{A_j}$  se sono order preserving o meno

- $Map_{A_j}$  è order-preserving: la traduzione considera tutte le partizioni di  $A_i$  e identifica tutte le partizioni di  $A_j$  che soddisfano la condizione di ordine. Formalmente:

$$Map_{cond}(A_i < A_j) \implies \vee_{p \in \text{partition}(A_i)} (I_i = ident_{A_i}(p) \wedge I_j \geq Map_{A_j}(p.\text{low}))$$

- $Map_{A_i}$  è order-preserving: simmetrico al caso precedente

$$Map_{cond}(A_i < A_j) \implies \vee_{p \in \text{partition}(A_j)} (I_j = ident_{A_j}(p) \wedge I_i \leq Map_{A_i}(p.\text{high}))$$

- $Map_{A_i}$  e  $Map_{A_j}$  sono entrambe order-preserving:

$$Map_{cond}(A_i < A_j) \implies \vee_{\phi} (Map_{A_i}(p_k.\text{low}) \leq Map_{A_j}(p_l.\text{high}))$$

dove  $\phi$  è  $p_k \in \text{partition}(A_i)$  e  $p_l \in \text{partition}(A_j)$

- $Map_{A_i}$  e  $Map_{A_j}$  sono entrambe random: la traduzione considera tutte le coppie di partizioni di  $A_i$  e  $A_j$  che soddisfano la condizione. Formalmente:

$$Map_{cond}(A_i < A_j) \implies \vee_{\phi} (I_i = ident_{A_i}(p_k) \wedge I_j = ident_{A_j}(p_l))$$

dove  $\phi$  è  $p_k \in partition(A_i)$ ,  $p_l \in partition(A_j)$  e  $p_l.high \geq p_k.low$

Ogni query Q nel database viene tradotta in:

- una query per il server  $Q_s$
- una per il client  $Q_c$

. La query per il server  $Q_s$  opera sugli indici quindi è la traduzione delle query per il client usando le regole viste sopra.

Quando il risultato arriva al client viene decriptato e il client può eseguire la propria query, rimuovendo le parti che non servono e che sono finite lì a causa della collisione tra indici.

L'importante è che il risultato sia completo: meglio avere più tuple piuttosto che meno.

Altra cosa a cui bisogna stare attenti: se il server ha anche funzioni di computazione devo fare in modo che parte della computazione sia a carico suo, non avrebbe senso portarsi sul client tutta una parte di db e poi fare tutta la computazione.

Vediamo un esempio:

Accounts			Accounts <sup>k</sup>				
Account	Customer	Balance	Counter	Etuple	I <sub>A</sub>	I <sub>C</sub>	I <sub>B</sub>
Acc1	Alice	100	1	x4Z3tfX2ShOSM	$\pi$	$\alpha$	$\mu$
Acc2	Alice	200	2	mNHg1oC010p8w	$\sigma$	$\alpha$	$\kappa$
Acc3	Bob	300	3	WslaCvtyF1Dxw	$\xi$	$\delta$	$\theta$
Acc4	Chris	200	4	JpO8eLTVgwV1E	$\rho$	$\alpha$	$\kappa$
Acc5	Donna	400	5	qctG6XnFNDTQc	$\varsigma$	$\beta$	$\kappa$
Acc6	Elvis	200	6	4QbqC3hxZHkIU	$\iota$	$\beta$	$\kappa$

La query originale avrà una struttura simile:

```
SELECT *
FROM Accounts
WHERE Balance = 200
```

e andrà ad ottenere come risultati Acc2, Acc4, Acc6.

La query tradotta  $Q_s$  invece avrà una struttura simile:

```
SELECT Etuple
FROM Accounts2k
WHERE Ib = k
```

che otterrà come risultati le righe 2, 4, 5, 6.

Ovviamente in questa query ci sono risultati che io non vorrei, quindi lato del DBMS (client) devo ripulire questi risultati sporchi e fare un'ultima query:

```
SELECT *
FROM Decrypt(Qs, Key)
WHERE balance = 200
```

che darà come risultato le righe Acc2, Acc4, Acc6.

### 5.1.3 Hash-based index

Basato sul concetto di one-way hash function.

Per ogni attributo  $A_i$  c'è una funzione di hash  $h : D_i B_i$  dove  $B_i$  è il dominio dell'index  $I_i$  associato ad  $A_i$ .

Data una tupla con il testo in chiaro  $t$  in  $r$ , il valore dell'indice che corrisponde a  $t[A_i]$  è  $h(t[A_i])$

Le proprietà importanti di ogni funzione sicura di hash  $h$  sono:

- **determinismo:**  $\forall x, y \in D_i : x = y \implies h(x) = h(y)$
- **collisione:** dati due valori  $x, y \in D_i$  con  $x \neq y$  possiamo avere che  $h(x) = h(y)$
- **strong mixing:** dai due valori vicini ma distinti  $x, y (|x - y| < \epsilon)$  scelto a caso in  $D_i$  la distribuzione della probabilità discreta della differenza  $h(x) - h(y)$  è uniforme

Accounts			Accounts <sup>†</sup>			
Account	Customer	Balance	Enc_tuple	I <sub>A</sub>	I <sub>C</sub>	I <sub>B</sub>
Acc1	Alice	100	x4Z3tX2ShOSM	$\pi$	$\alpha$	$\mu$
Acc2	Alice	200	mNHg1oC010p8w	$\sigma$	$\alpha$	$\kappa$
Acc3	Bob	300	WslaCvfyF1Dxw	$\xi$	$\delta$	$\theta$
Acc4	Chris	200	JpO8eLTvgwV1E	$\rho$	$\alpha$	$\kappa$
Acc5	Donna	400	qctGeXnFNDTQc	$\varsigma$	$\beta$	$\kappa$
Acc6	Elvis	200	4QbqC3hxZHkIU	$\iota$	$\beta$	$\kappa$

Abbiamo che:

$$h_c(Alice) = h_c(Chris) = \alpha$$

$$h_c(Donna) = h_c(Elvis) = \beta$$

$$h_c(Bob) = \delta$$

$$h_b(200) = h_b(400) = \kappa$$

...

Questo metodo di indexing supporta le query dove le condizioni sono formule booleane dei termini, nella forma:

- Attribute = Value
- Attribute1 = Attribute2, se Attribute1 e Attribute2 sono indicizzati con la stessa funzione di hashing

Non supporta query di range a causa dello strong mix.

La traduzione funziona come nel metodo partition-based.

### 5.1.4 Interval-based queries

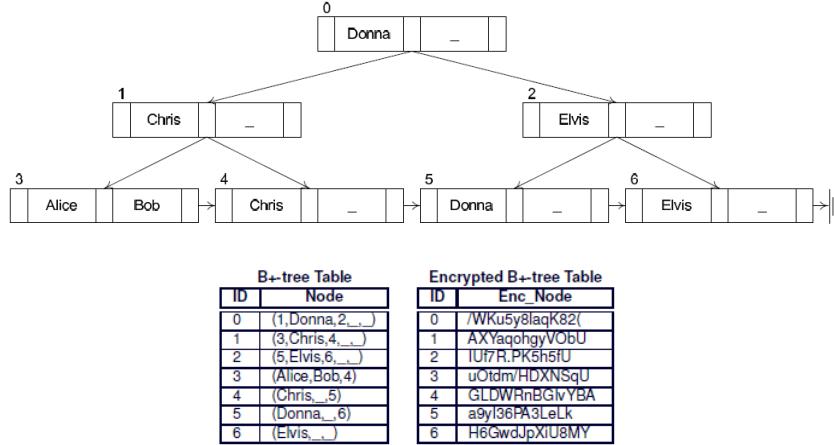
Le tecniche che mantengono l'ordine negli indici supportano le query basate sugli intervalli ma allo stesso modo sono esposte agli attacchi di inferenza: comparando le sequenze ordinate di testi e indici potremmo ricostruire la corrispondenza.

Le tecniche che non mantengono l'ordine non supportano le query basate sugli intervalli ma sono protette rispetto agli attacchi di inferenza.

I DBMSs supportano le query ad intervalli usando alberi B+, ma l'albero B+ definito dal server sugli indici non serve. Abbiamo due possibili soluzioni:

- calcolare i nodi nell'albero B+ al client e criptare ogni nodo come un intero al server

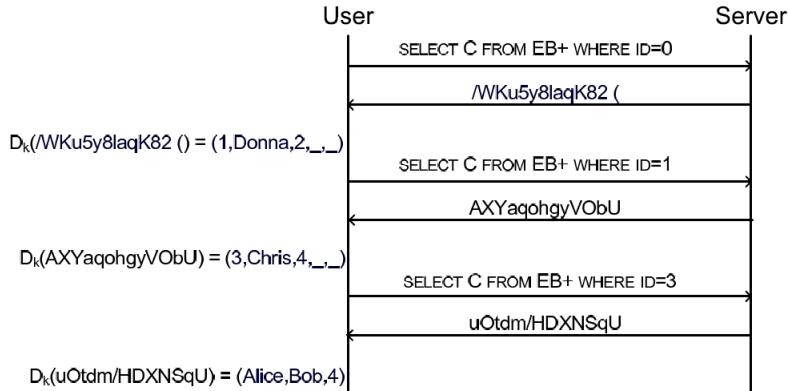
- L'attraversamento degli alberi B+ devono essere fatti lato trusted front-end



Proviamo a fare una query su questo schema:

```
SELECT * FROM Accounts WHERE Customer = "Bob"
```

Che viene interpretata ed eseguita così:



## 5.2 Searchable encryption

### 5.2.1 Order preserving

- **Order Preserving Encryption Schema (OPES)** prende in input una distribuzione target di valori di indici e applica una trasformazione che tiene conto dell'ordine in modo che i risultanti valori di indici seguano la distribuzione target. Vantaggi: la comparazione può essere effettuata direttamente sui dati criptati e la valutazione di query non produce risultati spuri. Svantaggi: è vulnerabile ad attacchi di inferenza.
- **Order Preserving Encryption with Splitting and Scaling (OPESS)** lo schema crea valori di indici tali che la loro distribuzione di frequenza sia piatta.

### 5.2.2 Fully Homomorphic Encryption

Questo metodo:

- ci permette di eseguire specifiche computazioni sui dati crittati
- la decrittazione del risultato della computazione restituisce il risultato delle operazioni eseguite sui dati con il testo in chiaro.

Recenti avanzamenti hanno prodotto un functional-encryption schema che tiene insieme diversi schemi esistenti (homomorphic, garbled circuit ...), ma continua ad essere computazionalmente complesso per essere applicato ad un effettivo DBMS

## 5.3 Inference Exposure

Ci sono due conflitti di requisiti nell'indicizzazione dei dati:

- gli indici devono fornire un meccanismo efficace di esecuzione delle query
- gli indici non dovrebbero permettere attacchi di inferenza o linking

È importante quindi misurare quantitativamente il livello di esposizione dovuto alla pubblicazione degli indici:

$$\epsilon = \text{Coefficiente di Esposizione}$$

La computazione del coefficiente di esposizione dipende sostanzialmente da due fattori:

- il metodo di indicizzazione applicato:
  - crittazione diretta
  - hashing
- la conoscenza che un intuso potrebbe avere a priori:
  - **Freq +  $DB^k$** : la distribuzione di frequenza dei valori in chiaro nel database originale (Freq), il database crittato ( $DB^k$ )
  - **DB +  $DB^k$** : il database in chiaro (DB) e il database crittato  $DB^k$

Le inferenze a cui possiamo essere soggetti sono:

**Freq +  $DB^k$ :**

- plaintext content: si può determinare l'esistenza di una certa tupla (o associazione di valori) nel db originale
- indexing function: si può determinare la corrispondenza tra indici e valori in chiaro

**DB +  $DB^k$ :**

- indexing function: si può determinare la corrispondenza tra indici e valori in chiaro

Andremo ad affrontare delle tecniche per calcolare il coefficiente di esposizione:

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

### 5.3.1 Quotient table

Vediamo un esempio di Freq +  $DB^k$ : date queste conoscenze pregresse

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

Accounts <sup>k</sup>						
Counter	Etuple	I <sub>A</sub>	I <sub>C</sub>	I <sub>B</sub>		
1	x4Z3tX2ShOSM	$\pi$	$\alpha$	$\mu$		
2	mNHg1oC010p8w	$\sigma$	$\alpha$	$\kappa$		
3	WslaCvtyF1Dxw	$\xi$	$\beta$	$\eta$		
4	JpO8eLTVgwV1E	$\rho$	$\gamma$	$\kappa$		
5	qctG6XnFNDTQc	$\varsigma$	$\delta$	$\theta$		
6	4QbqC3hxZHklU	$\iota$	$\varepsilon$	$\kappa$		

Le inferenze che possiamo fare sono:

- $I_A = \text{Account}$
- $I_C = \text{Customer}$
- $I_B = \text{Balance}$
- $\kappa = 200$  (indexing inference)
- $\alpha = \text{Alice}$  (indexing inference)
- $< \text{Alice}, 200 >$  è nella tabella (association inference)
- Alice è associata anche con un valore diverso da 200 (100,300,400, tutti equiprobabili)

Cosa può espormi? Il fatto di essere particolare (Alice è l'unica che appare due volte)

Cosa può proteggermi? Il fatto che tutti gli altri siano come me.

La corrispondenza tra un indice e il valore in chiaro può essere determinata quindi dall'occorrenza dell'indice o del valore. La protezione più basica è quella di avere valori con lo stesso numero di occorrenza.

La valutazione dell'esposizione dell'indice è basata su una relazione di equivalenza dove indice/valore in chiaro con lo stesso numero di occorrenza appartengono alla stessa classe: l'esposizione dei valori nelle classi di equivalenza C è  $\frac{1}{|C|}$

$$\begin{aligned}
A.1 &= \{\pi, \varpi, \xi, \rho, \varsigma, \iota\} = \{Acc1, \dots, Acc6\} \\
C.1 &= \{\beta, \gamma, \delta, \varepsilon\} = \{Bob, Chris, Donna, Elvis\} \\
C.2 &= \{\alpha\} = \{Alice\} \\
B.1 &= \{\mu, \eta, \theta\} = \{100, 300, 400\} \\
B.3 &= \{\kappa\} = \{200\}
\end{aligned}$$

INDEX_VALUES			QUOTIENT			INVERSE CARDINALITY		
$I_A$	$I_C$	$I_B$	$qt_A$	$qt_C$	$qt_B$	$ic_A$	$ic_C$	$ic_B$
$\pi$	$\alpha$	$\mu$	A.1	C.2	B.1	1/6	1	1/3
$\varpi$	$\alpha$	$\kappa$	A.1	C.2	B.3	1/6	1	1
$\xi$	$\beta$	$\eta$	A.1	C.1	B.1	1/6	1/4	1/3
$\rho$	$\gamma$	$\kappa$	A.1	C.1	B.3	1/6	1/4	1
$\varsigma$	$\delta$	$\theta$	A.1	C.1	B.1	1/6	1/4	1/3
$\iota$	$\varepsilon$	$\kappa$	A.1	C.1	B.3	1/6	1/4	1

$$\mathcal{E} = \frac{1}{n} \sum_{i=1}^n \prod_{j=1}^k IC_{i,j} = 1/18$$

Come possiamo vedere dall'immagine abbiamo diverse sigle i cui nomi sono facilmente riconducibili all'attributo e alla sua occorrenza (A.1 attributo A che compare solo una volta, C.2 attributo C che compare due volte ecc...).

Il procedimento è il seguente:

1. creo la tabella con i valori degli indici
2. mappo la tabella degli indici in una nuova tabella di quozienti
3. applicando la formula  $\frac{1}{|C|}$  ad ogni cella ottengo l'ultima tabella che rappresenta l'inverso della cardinalità. Da questa possiamo essere in grado di capire quali valori sono esposti e quali no, ad esempio i valori che hanno cardinalità inversa ad 1 sono esposti, mentre con  $\frac{1}{4|}$  ho il 25% di possibilità di avere quel valore.

L'ultima formula invece rappresenta l'esposizione totale della tabella:

- 1..n sono le righe della tabella
- 1..k sono le colonne
- il calcolo è la somma dei prodotti dei valori di ogni riga

### 5.3.2 RCV Graph

L'obiettivo è quello di proteggere il mapping tra indici e valori in chiaro.

Il modello è rappresentato da un grafo colorato indiretto (non c'è verso negli archi) di tipo RCV (riga, colonna, valore) dove:

- c'è un vertice di colore "colonna" per ogni attributo
- c'è un vertice di colore "riga" per ogni tupla
- c'è un vertice per ogni valore distinto in una colonna
- un arco connette ogni valore alla colonna e alla riga nella quale compare

RCV sui valori in chiaro è identico a quello sugli indici.

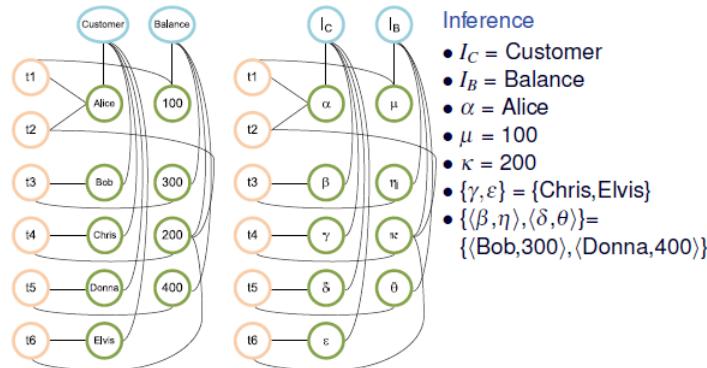
Il livello di esposizione può essere misurato valutando l'automorfismo del grafo.

Non è sufficiente contare il numero di automorfismi del grafo, infatti, se ci sono K automorfismi e in k di questi l'etichetta assegnata a v è la stessa, c'è una probabilità di  $\frac{k}{K}$  di identificare il valore.

Vediamo un esempio:

Customer	Balance	$I_C$	$I_B$
Alice	100	$\alpha$	$\mu$
Alice	200	$\alpha$	$\kappa$
Bob	300	$\beta$	$n$
Chris	200	$\gamma$	$\kappa$
Donna	400	$\delta$	$\theta$
Elvis	200	$\varepsilon$	$\kappa$

che viene rappresentata come:



L'insieme degli automorfismi forma un gruppo che è definito da una equitable partition, cioè la partizione dei nodi che si possono scambiare tra di loro (un nodo vale l'altro). Ogni sottoinsieme in una partizione è insieme con i nodi che si possono scambiare.

A questo punto posso utilizzare un algoritmo Nauty per derivare qualche partizione. Anche in questo caso la probabilità di individuare un elemento della partizione è  $\frac{1}{|C|}$ .

Rifacendoci all'esempio di prima:

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

Ho 9 elementi totali e 6 di confusione, quindi il mio coefficiente di esposizione è:  $= \frac{6}{9} = \frac{2}{3}$

## 5.4 Bloom filter

Il bloom filter è alla base della costruzione di alcune delle tecniche di indexing. È un metodo efficiente per codificare set membership (un dizionario su una struttura molto piccola). Gli elementi che lo definiscono sono:

- insieme di n elementi dove n è un numero molto alto

- vettore di  $l$  bits dove  $l$  è un numero piccolo (mi serve per mappare gli elementi)
- $h$  funzioni di hash indipendenti  $H_i : \{0, 1\}^* \rightarrow [1, l]$

Per inserire un elemento  $x$  setto ad uno i bit alle posizioni degli indici  $H_1(x), H_2(x) \dots, H_h(x)$   
 Per cercare un elemento  $x$  calcolo  $H_1(x), H_2(x) \dots, H_h(x)$  e controllo quale di quei valori sono settati nei bit del vettore.

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

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

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

Bloom filter con una sola funzione di hashing equivale all'hashing ordinario: occupano poco spazio (space efficient), ma gli elementi non possono essere rimossi.  
 Contiene falsi positivi (come abbiamo visto nell'esempio): nella teoria sono inaccettabili, ma nelle applicazioni pratiche è il prezzo da pagare per ottenere la space efficiency.

## 5.5 Integrità dei Dati

L'integrità dei dati si focalizza principalmente su due aspetti:

- **integrità in storage:** i dati devono essere protetti da modifiche (le modifiche non autorizzate devono essere tracciate)
- **integrità nel calcolo delle query:** i risultati delle query devono essere corretti e completi (dobbiamo tracciare anche errori e mancamenti da parti del server nel calcolo di query)

Per garantire l'integrità dei dati in storage possiamo optare per le firme digitali che vengono calcolate solitamente a livello di tuple:

- le firme di tabelle e attributi possono essere verificate solo dopo il download della tabella/colonna
- firmare a livello di cella causa un alto tasso di verifiche

Il costo della verifica cresce linearmente con il numero di tuple nel risultato della query: le firme di un insieme di tuple possono essere combinate per generare firme aggregate

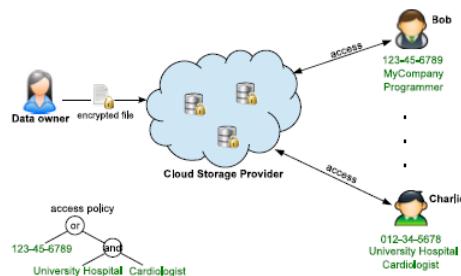
## 5.6 Selective Encryption and Over-Encryption

Siamo sempre nella situazione in cui uso un ente terzo per memorizzare i dati e voglio permettere ad altri di accedere ai dati. Voglio avere la possibilità di eseguire query senza dare al server l'accesso ai dati in chiaro. Come si può fare? Faccio query direttamente sul criptato (crittografia diretta o omomorfica) oppure sfrutto gli indici, che però mi danno meno protezione.

Un'altra caratteristica che potrei volere è quella di differenziare gli utenti e con loro anche le viste nei dati pubblicati. Questo vorrebbe dire potenziare le policy di controllo di accesso che richiedono al data owner di mediare le richieste di accesso (soluzione impraticabile se non inapplicabile) oppure rafforzare le autorizzazioni che non dovrebbero essere delegate al provider (il data owner dovrebbe rimanere in pieno controllo dei suoi dati).

Abbiamo due tipi di approcci:

- **Attribute-based encryption (ABE)**: permette le derivazioni di una chiave solo da parte di utenti che possiedono certi attributi (basato su crittografia asimmetrica)



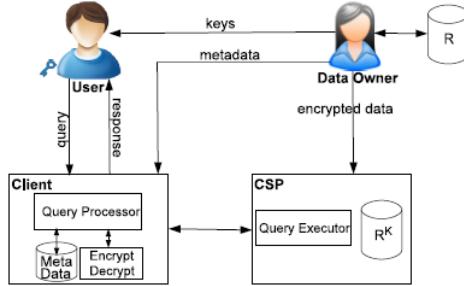
- **Selective encryption**: le policy di autorizzazione definite dal data owner vengono tradotte in policy di criptazione equivalenti



Le idee che stanno alla base sono:

- i dati devono fare da soli il controllo sugli accessi
- dobbiamo usare chiavi differenti per criptare i dati
- l'autorizzazione per accedere a una risorsa viene tradotta in conoscenza delle chiavi con le quali le risorse sono criptate

- ad ogni utente vengono comunicate solo le chiavi necessarie a decriptare i dati e le risorse a cui può avere accesso.

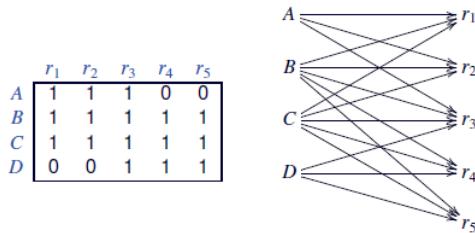


Il data owner definisce una policy di controllo di accesso discrezionale per regolare l'accesso alle risorse.

Una policy di autorizzazione A è un set di permessi nella forma  $\langle user, resource \rangle$  e può essere rappresentata come:

- una matrice di accesso
- un grafo diretto che ha un vertice per ogni utente u e ogni risorsa r, e archi da u a r per ogni permesso  $\langle u, r \rangle$

Diverse ACLs implicano anche diverse chiavi di criptazione



Come abbiamo detto in precedenza ogni policy di autorizzazione viene tradotta in un'equivalente policy di criptazione.

Abbiamo diverse possibili soluzioni:

- cripto ogni risorsa con una chiave diversa e consegno ad ogni utente le chiavi delle risorse a cui può accedere (richiede agli utenti di gestire tante chiavi quanti i numeri di risorse a cui sono autorizzati ad accedere)
- usare un metodo di derivazione delle chiavi per permettere agli utenti di derivare dalle chiavi tutte le altre a cui possono avere accesso (posso quindi dare una sola chiave ad ogni utente)

Abbiamo citato il metodo di derivazione delle chiavi, andiamo ad approfondirlo. Questo è basato su una gerarchia di derivazione delle chiavi ( $K, \preceq$ ) dove K è l'insieme di chiavi nel sistema, e  $\preceq$  è l'ordine parziale definito su K

La conoscenza della chiave del vertice  $v_1$  e di una parte dell'informazione disponibile pubblicamente permette il calcolo della chiave di un livello inferiore  $v_2$  tale che  $v_2 \preceq v_1$ .

$(K, \preceq)$  può essere rappresentato tramite un grafo dove ogni nodo rappresenta una chiave  $x$  nell'insieme  $K$  ed esiste l'arco da  $x$  a  $y$  sse  $y \preceq x$

In base alla funzione di ordine parziale definita su  $K$ , la gerarchia di derivazione delle chiavi può essere:

- una catena
- un albero
- un DAG

Le chiavi vengono arbitrariamente assegnate ai vertici. Viene associata anche un'etichetta  $l$  ad ogni chiave  $k$ . Una parte dell'informazione pubblica  $t_{i,j}$  chiamata **token** è associata ad ogni arco nella gerarchia.

Dato un arco  $(k_i, k_j)$ , il token  $t_{i,j}$  è calcolato come  $k_j \oplus h(k_i, l_j)$  dove:

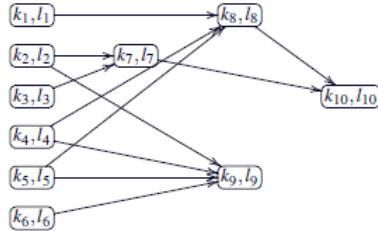
- $\oplus$  è l'operatore xor n-ario
- $h$  è una funzione di hash sicura

I vantaggi principali di utilizzare i tokens:

- sono pubblici e permettono agli utenti di derivare chiavi di criptazione multiple, preoccupandosi di gestirne solo una
- possono essere salvati in un server remoto (come i dati criptati) così che ogni utente possa accederci

Le relazioni tra le chiavi e i token possono essere rappresentati tramite un grafo **key and token** dove abbiamo:

- un vertice  $\langle k, l \rangle$ , dove  $k \in K$  è una chiave e  $l \in L$  è la corrispondente etichetta
- un arco da un vertice  $\langle k_i, l_i \rangle$  al vertice  $\langle k_j, l_j \rangle$  se esiste un token  $t_{i,j} \in T$  che permette la derivazione di  $k_j$  a partire da  $k_i$



Nell'esempio riportato sopra conoscendo  $k_1$  posso derivare  $k_8$  e  $k_{10}$

Come abbiamo detto in precedenza dobbiamo tradurre le policy di autorizzazione in policy di criptazione. Assunzioni iniziali:

- ad ogni utente deve essere rilasciata una sola chiave
- ogni risorsa è criptata una sola volta (con una sola chiave)

Abbiamo quindi una funzione  $\phi : U \cup RL$  che descrive le associazioni tra utenti e (l'etichetta della loro) chiave e l'associazione tra risorse e (l'etichetta della) chiave usata per criptarla.

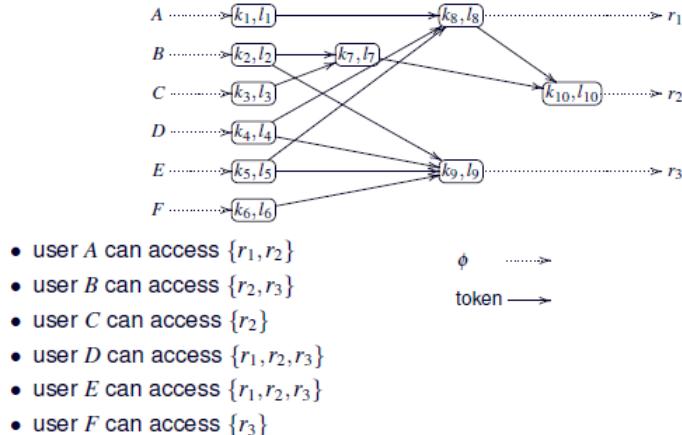
Definiamo formalmente policy di criptazione:

una policy di criptazione sugli utenti  $U$  e risorse  $R$ , denotata con  $\epsilon$ , è una sestupla  $\langle U, R, K, L, \phi, t \rangle$  dove:

- $K$  è l'insieme delle chiavi definite nel sistema e  $L$  è l'insieme di etichette corrispondenti
- $\phi$  è un assegnamento di una chiave e uno schema di criptazione
- $T$  è un insieme di token definito su  $K$  e  $L$

Anche questo tipo di policy può essere rappresentata attraverso un grafo estendendo quello visto precedentemente per includere:

- un vertice per ogni utente e ogni risorsa
- un arco che collega ogni utente  $u$  al vertice  $\langle k, l \rangle$  tali che  $\phi(u) = l$
- un arco che collega ogni vertice  $\langle k, l \rangle$  ad ogni vertice delle risorse tali che  $\phi(r) = l$



L'obiettivo della traduzione è quello di tradurre una policy di autorizzazione  $A$  in una policy di criptazione equivalente  $\epsilon$ .

$A$  ed  $\epsilon$  sono equivalenti se garantiscono gli stessi accessi, senza garantire di più o di meno.

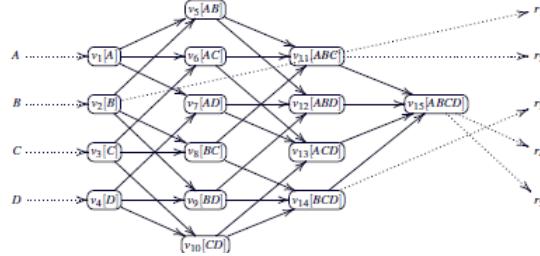
Questa traduzione può essere eseguita in due modi:

- approccio naive: ogni utente è associato ad una chiave differente, ogni risorsa è associata con una chiave diversa, un token  $t_{u,r}$  è generata e pubblicata per ogni permesso  $\langle u, r \rangle$ , ma produrre e gestire un token per ogni permesso può essere infattibile praticamente

- creare ACL: creo gruppi di utenti con gli stessi privilegi e cripto risorse con chiave data a tutto il gruppo

È possibile creare un grafo di policy di criptazioni sfruttando la gerarchia tra insiemi di utenti indotti dalla relazione di ordine parziale.

Se il sistema ha un grande numero di utenti la policy di criptazione ha un grande numero di token e chiavi ( $2^{|U|-1}$ )

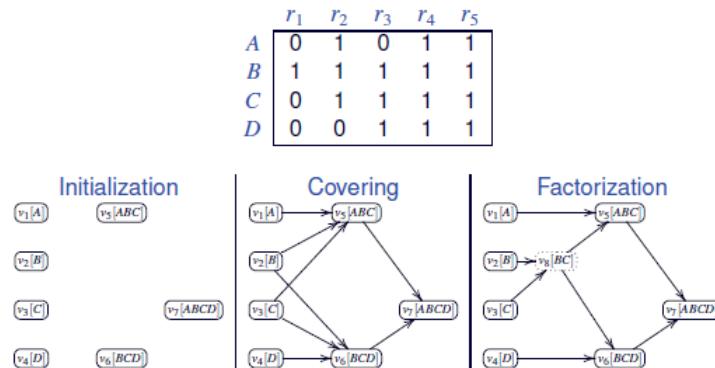


Possiamo notare come i gruppi di utenti che non corrispondono a nessun ACL non abbiano nessuna chiave. L'obiettivo è quindi quello di calcolare la policy di criptazione minima che minimizza il numero di token che deve essere mantenuto dal server. Le possibili soluzioni sono algoritmi euristici basati su osservazioni:

- solo i vertici associati a gruppi di utenti che corrispondono a delle ACL devono essere associate ad una chiave
- il grafo delle policy di criptazione può includere solo i verci che servono per imporre una certa policy di autorizzazione, connettendoli per garantire una corretta derivazione di chiavi
- altri vertici possono essere inclusi se sono utili per ridurre la dimensione del catalogo

Per costruire un grafo di chiavi e token abbiamo 3 fasi:

1. **Inizializzazione:** creo un vertice/chiave per ogni utente e per ogni ACL non singleton
2. **Copertura:** per ogni vertice v corrispondente a un ACL non singleton troviamo una copertura senza ridondanza (deve coprire l'insieme)
3. **Fattorizzazione:** fattorizzare gli antenati comuni



A questo punto la mia policy di criptazione è decisa. Dal token graph (l'ultimo a destra) posso dare a ciascun utente la sua chiave e criptare ogni risorsa con la sua chiave, creando così la tabella dei token:

$u$	$\phi(u)$	$r$	$\phi(r)$	source	destination	token_value
A	$v_1.l$	$r_1$	$v_2.l$	$v_1.l$	$v_5.l$	$t_{1,5}$
B	$v_2.l$	$r_2$	$v_5.l$	$v_2.l$	$v_8.l$	$t_{2,8}$
C	$v_3.l$	$r_3$	$v_6.l$	$v_3.l$	$v_8.l$	$t_{3,8}$
D	$v_4.l$	$r_4, r_5$	$v_7.l$	$v_4.l$	$v_6.l$	$t_{4,6}$
				$v_5.l$	$v_7.l$	$t_{5,7}$
				$v_6.l$	$v_7.l$	$t_{6,7}$
				$v_8.l$	$v_5.l$	$t_{8,5}$
				$v_8.l$	$v_6.l$	$t_{8,6}$

Quando proprietari diversi hanno bisogno di condividere i loro dati, l'uso di una chiave permette a due proprietari di condividere una chiave segreta per seguenti usi crittografici.

Quando si autorizzano cambiamenti dinamici, il proprietario dei dati deve:

- scaricare le risorse dal server
- creare una nuova chiave per le risorse
- decriptare la risorsa con la vecchia chiave
- criptare le risorse con la nuova chiave
- caricare le risorse sul server e comunicare i cambiamenti

Possiamo notare come questo procedimento sia davvero inefficiente e per questo subentra **over-encryption**.

Il funzionamento è semplice: le risorse sono criptate due volte:

- dal proprietario con una chiave condivisa con gli utenti e sconosciuta al server (Base Encryption Layer - BEL level)
- dal server, con una chiave condivisa con gli utente autorizzati (Surface Encryption Layer - SEL level)

Per accedere ad una risorsa un utente deve sapere entrambe le chiavi BEL e SEL. Garantire e revocare le operazioni può richiedere l'aggiunta di nuovi token a BEL level e l'aggiornamento del SEL level in conseguenza alle operazioni eseguite.

A livello BEL possiamo distinguere due chiavi, una di accesso  $k_a$  e una di derivazione (k):

- ogni nodo in Bel è associato con una coppia di chiavi  $(k, k_a)$  dove  $k_a = h(k)$  dove  $h$  è una funzione di hash, e una coppia di  $(l, l_a)$
- la chiave k con etichetta l è usata per scopi di derivazione
- la chiave  $k_a$  con etichetta  $l_a$  è usata per criptare le risorse associate al nodo
- questa distinzione separa i due ruoli associati alle chiavi: abilita la derivazione della chiave e abilita l'accesso alla risorsa

Il livello SEL è caratterizzato da una policy di criptazione definita come illustrato precedentemente. Esistono due tipi di SEL:

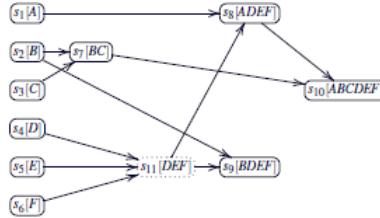
- **Full\_SEL**: parte da una SEL identica alla BEL e tiene SEL sempre aggiornata per rappresentare la policy corrente (tutte le risorse sono criptate)
- **Delta\_SEL**: parte da una SEL vuota e aggiunge elementi man mano che la policy si evolve, in modo che la coppia BEL-SEL rappresenti la policy (server critta solo quando serve coprire qualcosa, se non cambia la politica non ha senso cambiare ancora)

Vediamo un esempio:

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

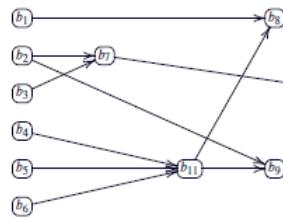


BEL

$u$	$\phi_b(u)$	$r$	$\phi_b(r)$
A	$b_1.l$	$r_1, r_2$	$b_4.l_a$
B	$b_2.l$	$r_3, r_4, r_5$	$b_7.l_a$
C	$b_3.l$	$r_6, r_7$	$b_8.l_a$
D	$b_4.l$	$r_8$	$b_9.l_a$
E	$b_5.l$	$r_9$	$b_{10}.l_a$
F	$b_6.l$		

Full\_SEL

$u$	$\phi_s(u)$	$r$	$\phi_s(r)$
A	$s_1.l$	$r_1, r_2$	$s_4.l$
B	$s_2.l$	$r_3, r_4, r_5$	$s_7.l$
C	$s_3.l$	$r_6, r_7$	$s_8.l$
D	$s_4.l$	$r_8$	$s_9.l$
E	$s_5.l$	$r_9$	$s_{10}.l$
F	$s_6.l$		



BEL				Delta_SEL			
$u$	$\phi_b(u)$	$r$	$\phi_b(r)$	$u$	$\phi_s(u)$	$r$	$\phi_s(r)$
A	$b_1.l$	$r_1, r_2$	$b_4.l_a$	A	$s_1.l$	$r_1, \dots, r_9$	NULL
B	$b_2.l$	$r_3, r_4, r_5$	$b_7.l_a$	B	$s_2.l$		
C	$b_3.l$	$r_6, r_7$	$b_8.l_a$	C	$s_3.l$		
D	$b_4.l$	$r_8$	$b_9.l_a$	D	$s_4.l$		
E	$b_5.l$	$r_9$	$b_{10}.l_a$	E	$s_5.l$		
F	$b_6.l$			F	$s_6.l$		

The diagram illustrates the relationships between users (b1 to b6) and resources (r1 to r10). The connections are as follows:

- b1 connects to b8.
- b2 connects to b7.
- b3 connects to b7 and b10.
- b4 connects to b11.
- b5 connects to b11.
- b6 connects to b11.
- b7 connects to b8.
- b8 connects to b10.
- b11 connects to b9.

Le evoluzioni di BEL e SEL sono gestite da:

- procedure over-encrypt (lato SEL) che regolano il processo di aggiornamento attraverso l'over-encrypting delle risorse a livello SEL. Il server riceve una richiesta per sovrascrivere una risorsa per un gruppo di utenti (cioè per sovrascrivere una risorsa così da renderla disponibile solo per quel gruppo di utenti). Se la risorsa è già sovra criptata allora il server la apre e, se non esiste una chiave che corrisponde a un insieme di utenti, il server crea la chiave. Infine il server critta le risorse con la nuova chiave.
- garantisce e revoca le procedure che servono per garantire e revocare privilegi.
  - Nel caso del grant (lato BEL): voglio garantire ad un utente l'accesso a una risorsa che è crittata con una certa chiave. Quindi voglio aggiungere l'utente all'ACL della risorsa. Se l'utente non riesce a derivare la chiave, aggiungo un token per derivare quella chiave. Se ad esempio questo nuovo utente non può accedere alle risorse della chiave che gli è stata data, allora il server crea delle partizioni (in base all'ACL) e fa over encryption. A questo punto come faccio a fare in modo che il server sappia che il nuovo utente può accedere alla risorsa? Il server fa over encryption.
  - Nel caso del revoke (lato BEL): l'utente può accedere alla risorsa; l'obiettivo è togliergli l'accesso. Lato proprietario si aggiorna l'ACL. Il server deve restringere l'accesso alla risorsa facendo over encryption.

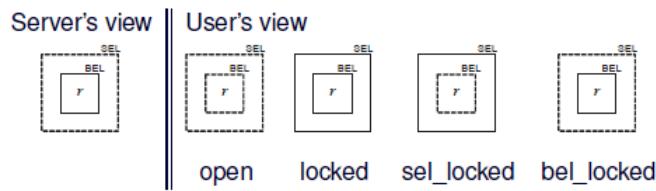
Le policy di criptazione BEL e SEL sono equivalenti alle policy di autorizzazione dichiarate durante la fase di inizializzazione. Le procedure viste poco fa preservano l'equivalenza, la funzione di derivazione delle chiavi che è stata adottata è sicura, tutte le funzioni di criptazione e i token sono robusti e non possono essere rotte. Ogni utente può maneggiare correttamente le sue chiavi senza la possibilità di poter rubare quelle di un altro utente.

Nonostante ciò si è vulnerabili agli attacchi di collusione: combinando le conoscenze due o più entità riescono a conoscere cose a cui nessuno di loro avrebbe accesso. Le collusioni possono essere tra utenti e col server. L'attacco dipende dalle diverse

viste che gli attaccanti hanno nei confronti di una certa risorsa r.  
 Abbiamo quattro possibili scenari:

- open: l'utente conosce le chiavi del BEL e del SEL
- locked: l'utente non conosce nessuna chiave
- sel\_locked: l'utente conosce solo la chiave del BEL
- bel\_locked: l'utente conosce solo la chiave del SEL

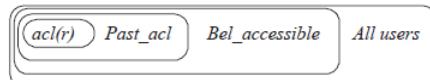
Il server ha sempre un tipo di vista bel\_locked.



Ogni livello è rappresentato come una ringhiera: discontinuo se la chiave è conosciuta, continuo se la chiave è sconosciuta.

Mi preoccupo delle collisioni in sel\_locked, cioè quando l'utente e il server sanno qualcosa e mettono insieme.

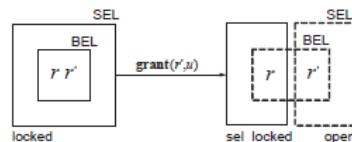
Consideriamo una risorsa r e la storia della sua  $\text{acl}(r)$ . Gli utenti presenti in questa storia possono essere classificati in quattro categorie:



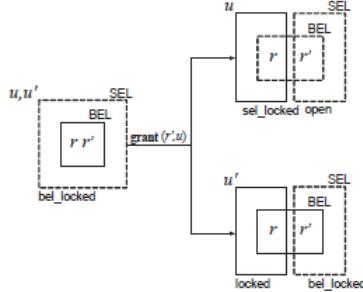
C'è rischio di collusione per r se ci sono utenti in Bel\_Accessible che non appartengono a Past\_acl (ha un accesso a cose a cui non ha mai avuto accesso).

Un utente può avere la vista sel\_locked su r se (chiedere la criptazione lato server) se:

- utente autorizzato e lo sto revocando
- risorsa coinvolta in policy split: l'utente ha accesso ad  $r'$  (non a r), criptato a livello BEL con la stessa chiave di r. Quindi devo chiedere la criptazione lato SEL per evitare che l'utente acceda anche ad r avendo la sua chiave.



Abbiamo la stessa cosa anche nel Delta\_SEL (sempre per un problema di policy split):



Quindi ricapitolando:

- in Full\_SEL
  - tra gli utenti non c'è problema, gli utenti non guadagnano mai nello scambio
  - col server è possibile che ci siano collusioni, ma sono facilmente identificabili ed evitabili tramite re-criptazione
- in Delta\_SEL ci sono problemi se lato utente al tempo 0 mi scarico tutto quello che viene messo fuori in un secondo momento posso usare la chiave che ho io per accedere alle risorse.

## 5.7 Frammentazione e criptazione

La criptazione rende la valutazione delle query e l'esecuzione delle applicazioni più costose o non sempre possibili.

Spesso la cosa più sensibile è l'associazione tra due valori di due attributi (più che i valori presi singolarmente) e per proteggere tale associazione si potrebbe romperla al posto che criptarla.

Abbiamo dei vincoli di confidenzialità che vengono specificati dall'utente e possono essere riferiti ai singoli attributi o alle associazioni. Per esempio dato  $R = (\text{Name}, \text{DoB}, \text{Gender}, \text{Zip}, \text{Position}, \text{Salary}, \text{Email}, \text{Telephone})$

- $\{\text{Telephone}\}, \{\text{Email}\}$ : sono sensibili e non possono essere salvati in chiaro
- $\{\text{Name}, \text{Salar}\}, \{\text{Name}, \text{Position}\}, \{\text{Name}, \text{DoB}\}$ : gli attributi Salary, Position, DoB sono privati di un individuo e non possono essere salvati in chiaro con l'associazione col nome
- $\{\text{DoB}, \text{Gender}, \text{Zip}, \text{Salary}\}, \{\text{DoB}, \text{Gender}, \text{Zip}, \text{Position}\}$ : gli attributi DoB, Gender, Zip potrebbero essere dei quasi identificatori
- $\{\text{Position}, \text{Salary}\}, \{\text{Salary}, \text{DoB}\}$ : le regole di associazione tra Position e Salary e tra Salary e DoB dovrebbero essere protette da un attaccante

### 5.7.1 Server non comunicanti

I vincoli di confidenzialità sono rinforzati dividendo le informazioni su due server indipendenti che non possono comunicare. Così facendo le associazioni sensibili sono protette distribuendo gli attributi tra i due server e la criptazione è applicata solo quando viene esplicitamente chiesto dai vincoli di confidenzialità o quando salvare un attributo in chiaro in uno dei due server potrebbe esporre almeno un'associazione sensibile.

Il vincolo di confidenzialità C definiti su una relazione R sono rafforzati dividendo R in  $\langle R_1, R_2, E \rangle$  dove:

- $R_1$  e  $R_2$  includono un tuple ID univoco necessario per garantire una divisione senza perdite
- $R_1 \cup R_2 = R$
- E è il set di attributi criptati e  $E \subseteq R_1$ ,  $E \subseteq R_2$
- per ogni  $c \in C$ ,  $c(R_1 - E)$  e  $c(R_2 - E)$

Per ricostruire i frammenti devo avere in comune qualcosa che è chiave, ad esempio un tuple ID. Inoltre, nessuno dei due frammenti contiene in chiaro attributi che fanno tutti parte di un vincolo di confidenzialità. Un vincolo di confidenzialità non è contenuto nella sua globalità in nessuno dei frammenti in chiaro.

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

$F_1$					$F_2$			
tid	Name	YoB	SSN <sup>k</sup>	Disease <sup>k</sup>	tid	Job	SSN <sup>k</sup>	Disease <sup>k</sup>
1	Alice	1980	jdks	hyaf4k	1	Clerk	uwq8hd	jsd7ql
2	Bob	1980	u9hs9	j97;qx	2	Doctor	j-0.dl;	0],nid
3	Carol	1970	j9und	9jp'md	3	Nurse	8ojqqdkf	j-0/?n
4	David	1970	p0vp8	p;nd92	4	Lawyer	j0i12nd	5lkdpq
5	Eva	1970	8nn[	0-mw-n	5	Doctor	m][9;'s	j0982e
6	Frank	1960	jsjMK	wqp9[i	6	Doctor	aQ14 [	jnd%d
7	Gary	1960	871'D	L0MB2G	7	Teacher	8qsdQW	OP[
8	Hilary	1960	8pm]n	@h8hwu	8	Nurse	0890UD	UP0D@

**Come si eseguono le query?** A livello logico rimpiazzo R con la join tra  $R_1 \bowtie R_2$ . Quindi faccio la query sul join, sfruttando il più possibile i server. Posso eseguire la query in due modi:

- eseguo le due query in parallelo, faccio il join e ho il risultato → se i server conoscono la query i miei dati sono esposti
- faccio la query da una parte e poi faccio un semi join con l'altra tabella sfruttando il tuple ID (esempio: estraggo le persone nate dopo il 2000, prendo il tuple ID di queste persone e utilizzo i tuple ID per andare a prendere i dati nell'altro frammento, così non devo scaricare tutti i dati) → comunico al server quali tuple ID mi servono

**Come faccio la frammentazione?** Applichiamo un approccio brute force per ottimizzare il carico di lavoro W:

- per ogni possibile decomposizione sicura di R:
  - ottimizzo ogni query in W per la decomposizione
  - stimo il costo totale per eseguire le query in W usando il piano di query ottimizzate
- seleziono la decomposizione che ha il minor costo totale

Questo però è un approccio troppo costoso.

Usiamo quindi la matrice di affinità: attributi sulle righe e sulle colonne, la cella  $i,j$  mi dice quante volte gli attributi  $i$  e  $j$  compaiono insieme nelle query. La diagonale principale mi dice quanto costa criptare quell'attributo. Per ottimizzare andrò a criptare e frammentare in modo tale da pagare il minor costo possibile quando andrò a fare le query.

Formalmente vogliamo minimizzare:

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

Questo problema è equivalente al problema di colorazione di un ipergrafo (un ipergrafo è una generalizzazione di un grafo dove gli archi possono collegare qualsiasi numero di vertici - ogni server ha un colore diverso) dove gli attributi sono vertici, la matrice di affinità mi dà il peso degli archi, la diagonale principale mi dà il peso dei vertici. I vincoli di confidenzialità C rappresentano un ipergrafo  $H(R,C)$  sugli stessi vertici.

Dobbiamo trovare la soluzione in modo che:

- nessun arco sia monocromatico
- il peso dei vertici bicromatici sia minimizzato
- un vertice può essere cancellato pagando il peso dei vertici

Anche in base alla formula di prima abbiamo che:

- il peso degli archi bicromatici è  $\sum_{i,j:i \in (R_1 - E), j \in (R_2 - E)} M_{i,j}$
- il peso della cancellazione di un vertice è  $\sum_{i \in E} M_{i,i}$

L'obiettivo è sempre quello di minimizzare la somma vista in precedenza.

Cosa ho alla fine: due frammenti, qualcosa in chiaro e qualcosa criptato.

Problemi della tecnica che si basa su server non comunicanti: muro che c'è fra i due server, l'assunzione di avere questo muro è un po'debole nel mondo odierno.

### 5.7.2 Frammenti non linkabili

L'accoppiata frammentazione/criptazione è interessante e porta molti vantaggi ma assumere che ci siano due server non comunicanti:

- è difficile da forzare nella realtà

- limita il numero delle associazioni che possono essere risolte frammentando i dati

Una frammentazione di R è un insieme di frammenti  $F = \{F_1, \dots, F_m\}$  dove  $F_i \subseteq R$  per  $i = 1, \dots, m$

Una frammentazione F di R rafforza correttamente un insieme di vincoli di confidenzialità C se le seguenti condizioni sono verificate:

- ogni singolo frammento soddisfa i vincoli
- i frammenti non hanno attributi in comune, altrimenti avrei la possibilità di fare linking

Ogni frammento F è mappato in un frammento fisico contenente:

- gli attributi nel frammento in chiaro
- tutti gli altri attributi di R criptati (viene applicato salt ad ogni criptazione)

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

$F_1$			$F_2$			$F_3$			
salt	enc	Name	YoB	salt	enc	Job	salt	enc	Disease
$S_{11}$	Bd6l3	Alice	1980	$S_{21}$	8de6TO	Clerk	$S_{31}$	ew3)VI	Asthma
$S_{12}$	Oij3X.	Bob	1980	$S_{22}$	X'mlE3	Doctor	$S_{32}$	LkEd69	Asthma
$S_{13}$	9kEf6?	Carol	1970	$S_{23}$	wq.vy0	Nurse	$S_{33}$	w8vd66	Asthma
$S_{14}$	ker5/2	David	1970	$S_{24}$	nh-l3a	Lawyer	$S_{34}$	1"qPdd	Bronchitis
$S_{15}$	C:mE91	Eva	1970	$S_{25}$	hn%kj	Doctor	$S_{35}$	(mz2eW	Bronchitis
$S_{16}$	4IDwqz	Frank	1960	$S_{26}$	,vf5eS	Doctor	$S_{36}$	wD)x1X	Gastritis
$S_{17}$	me3,op	Gary	1960	$S_{27}$	e4+YUp	Teacher	$S_{37}$	OopAuEl	Gastritis
$S_{18}$	zWf4g>	Hilary	1960	$S_{28}$	pgt6eC	Nurse	$S_{38}$	Sw@Fez	Diabetes

**Come si eseguono le query?** Non mi serve più andare a recuperare tutti i frammenti, me ne basta uno. Se la query coinvolge un attributo criptato potrebbe essere necessaria un'altra query lato client.

Se volessi trovare le malattie dei pazienti nati dopo il 1980 farei una query sul frammento dove c'è la data di nascita così da avere meno tuple da decriptare per avere la malattia.

**Come frammentiamo?** Come frammentiamo e soprattutto rispetto a cosa frammentiamo? L'obiettivo che vogliamo ottenere è quello di avere una frammentazione che mi permette di eseguire query efficienti. Possiamo frammentare in base a tre criteri:

- numero di frammenti
- affinità tra gli attributi

- query workload

Tutti i criteri devono rispettare il criterio di massima visibilità: solo gli attributi che appaiono in vincoli singleton (attributi sensibili) devono essere criptati, tutti gli altri attributi non sensibili devono avere almeno un frammento in cui siano in chiaro.

Quello che vogliamo fare è minimizzare la frammentazione. Per ridurre la complessità vogliamo determinare una frammentazione corretta ma col minor numero di frammenti.

Una frammentazione è minimale se tutte le frammentazioni che possono essere ottenute dalla frammentazione minimale unendo qualsiasi coppia di frammenti, viola almeno un vincolo.

MEDICALDATA						Confidentiality constraints						
SSN	Name	DoB	Zip	Illness	Physician	$c_0 = \{\text{SSN}\}$	$c_1 = \{\text{Name}, \text{DoB}\}$	$c_2 = \{\text{Name}, \text{Zip}\}$	$c_3 = \{\text{Name}, \text{Illness}\}$	$c_4 = \{\text{Name}, \text{Physician}\}$	$c_5 = \{\text{DoB}, \text{Zip}, \text{Illness}\}$	$c_6 = \{\text{DoB}, \text{Zip}, \text{Physician}\}$
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							

#### Minimal fragmentation $\mathcal{F}$

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

Unendo due dei frammenti si violerebbe almeno un vincolo.

Anche in questo caso ho una matrice di affinità. I frammenti vengono calcolati considerando gli attributi che più "vogliono" stare insieme (maximum affinity). Per calcolare maximum affinity si può usare un'euristica che ad ogni passo prende la mossa migliore.

Vediamo un esempio:

MEDICALDATA						Confidentiality constraints						
SSN	Name	DoB	ZIP	Illness	Physician	$c_0 = \{\text{SSN}\}$	$c_1 = \{\text{Name}, \text{DoB}\}$	$c_2 = \{\text{Name}, \text{ZIP}\}$	$c_3 = \{\text{Name}, \text{Illness}\}$	$c_4 = \{\text{Name}, \text{Physician}\}$	$c_5 = \{\text{DoB}, \text{ZIP}, \text{Illness}\}$	$c_6 = \{\text{DoB}, \text{ZIP}, \text{Physician}\}$
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							

$F_1 = \{n\}$	$F_1$	$F_2$	$F_3$	$F_4$	$F_5$	$n$	$x$	$x$	$x$	$x$	
$F_2 = \{d\}$	$F_2$		$10$	$5$	$25$	$d$	$x$			$x$	$x$
$F_3 = \{z\}$	$F_3$			$5$	$20$	$30$		$x$		$x$	$x$
$F_4 = \{\}$	$F_4$				$10$	$5$			$x$		$x$
$F_5 = \{p\}$	$F_5$					$15$				$x$	$x$

In quest'immagine possiamo vedere una tabella di medical data, dei vincoli di confidenzialità e delle frammentazioni di base. La matrice in basso a sinistra rappresenta

la matrice di affinità ed è scritta per metà poichè la metà sotto sarebbe uguale a quella sopra e la diagonale, non essendoci nulla di crittato, è cancellata. La matrice in basso a destra rappresenta gli attributi e i vari vincoli a cui sono legati.

	$F_1$	$F_2$	$F_3$	$F_4$	$F_5$		$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$
$F_1=\{n\}$	$F_1$	-1	-1	-1	-1		n	✓	✓	✓	✓	
$F_2=\{d\}$	$F_2$		5	20	30		d	✓			x	x
$F_3=\{z\}$	$F_3$			10	5		z		✓		x	x
$F_4=\{i\}$	$F_4$				15		i		✓		x	
$F_5=\{p\}$	$F_5$						p			✓		x

In questa foto possiamo vedere come tutta la prima riga della matrice dei frammenti sia stata messa a -1. Questo è avvenuto perchè l'attributo name (contenuto nel frammento  $F_1$ ), per i vincoli di confidenzialità, non può stare con nessun altro attributo (altrimenti violerebbe i vincoli). Mettendo a -1 il valore non andremo a considerarli successivamente.

Facendo la mossa localmente migliore (in un algoritmo con approccio greedy) so che otterrò un risultato che è globalmente migliore. Quindi in questo caso la mossa migliore è unire  $F_2$  ed  $F_5$  (visto che il loro incrocio ha valore 30) e facendo questo vuol dire che  $F_5$  scompare e dobbiamo regolare le affinità di  $F_2$ .

Le nuove affinità si calcolano come seguono:

- $F_3$  voleva stare 5 con  $F_5$ , quindi visto che ora  $F_5$  sta con  $F_2$ , aggiungiamo 5 all'incrocio  $F_2/F_3$
- $F_4$  voleva stare 15 con  $F_5$ , quindi visto che  $F_5$  sta con  $F_2$ , aggiungiamo 15 all'incrocio  $F_2/F_4$

	$F_1$	$F_2$	$F_3$	$F_4$	$F_5$		$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$
$F_1=\{n\}$	$F_1$	-1	-1	-1			n	✓	✓	✓	✓	
$F_2=\{d,p\}$	$F_2$		-1	35			d	✓			x	✓
$F_3=\{z\}$	$F_3$			10			z		✓		x	✓
$F_4=\{i\}$	$F_4$						i		✓		x	
$F_5=\{p\}$	$F_5$						p			✓		✓

Quello che succede è che dovrei avere l'incrocio  $F_2$  (DoB e Physician)  $F_3$  (ZIP) che vale 10, ma in realtà vale -1, perchè il vincolo  $c_6$  ci dice che questi tre attributi non possono stare insieme.

	$F_1$	$F_2$	$F_3$	$F_4$	$F_5$		$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$
$F_1=\{n\}$	$F_1$	-1	-1				n	✓	✓	✓	✓	
$F_2=\{d,p,i\}$	$F_2$		-1				d	✓		✓	✓	
$F_3=\{z\}$	$F_3$						z		✓		✓	✓
$F_4=\{i\}$	$F_4$						i		✓		✓	
$F_5=\{p\}$	$F_5$						p			✓		✓

Eseguendo una nuova iterazione otteniamo la situazione riportata sopra dove non possiamo più eseguire mosse avendo solo valori negativi.

L'affinità massima di frammentazione F vale 65. È la soluzione ottima poichè mettendo insieme due frammenti si violerebbe almeno un vincolo.

## 5.8 Frammentazione

### 5.8.1 Keep a few

L'idea che sta alla base è che:

- la criptazione rende il processo di esecuzione delle query più costoso e non sempre possibile
- la criptazione porta a dover gestire un numero grande di chiavi

Perchè non eliminare del tutto la criptazione?

Il problema è capire quanto posso caricare su server esterni, l'obiettivo infatti è tenere il meno possibile.

Devo determinare una frammentazione a due dove uno sono io (l'owner) e uno è il server.

Dati  $R(A_1, \dots, A_n)$  uno schema relazionale e  $C = \{c_1, \dots, c_m\}$  vincoli di confidenzialità su R, vogliamo determinare una frammentazione  $F = \langle F_O, F_S \rangle$  dov:

- $F_O \cup F_S = R$  (**completezza**)
- $\forall c \in C, cF_S$  (**confidenzialità**)
- $F_O \cap F_S = \emptyset$  (**non ridondanza**)

A livello fisico  $F_O$  e  $F_S$  hanno un attributo comune (tid) per permettere l'esecuzione di join senza perdite di dati.

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

$F_O$			$F_S$		
tid	SSN	Job	tid	Name	YoB
1	123456789	Clerk	1	Alice	1980
2	234567891	Doctor	2	Bob	1980
3	345678912	Nurse	3	Carol	1970
4	456789123	Lawyer	4	David	1970
5	567891234	Doctor	5	Eva	1970
6	678912345	Doctor	6	Frank	1960
7	789123456	Teacher	7	Gary	1960
8	891234567	Nurse	8	Hilary	1960

## 5.9 Valutazione di Query

Una query può avere dentro delle condizioni che possono coinvolgere attributi miei ( $C_O$ ), del server ( $C_S$ ) o di entrambi ( $C_{SO}$ ). Devo tradurre la mia query in query equivalenti che lavorano lato mio, lato server e combinato.

Usando come esempio i frammenti precedenti proviamo a capire la valutazione della query:

```

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

```

Le condizioni dello WHERE sono divise in:

- $(C_O) = \{\text{Disease} = \text{"Bronchitis"}\}$
- $(C_S) = \{\text{YoB} = \text{"1970"}\}$
- $(C_{SO}) = \{\text{Name} = \text{Job}\}$

Abbiamo diverse strategie da poter adottare:

- Server-Client strategy
  1. il server valuta  $(C_S)$  e ritorna il risultato al client
  2. il client riceve il risultato ed esegue la join con il suo frammento
  3. il client valuta  $(C_{SO})$  e  $(C_O)$  sul join
- Client-Server strategy
  1. il client valuta  $(C_O)$  e manda al server la serie di tid
  2. il server fa la join col suo frammento ed esegue la sua query ritornando il risultato al client
  3. il client esegue la join del suo frammento con il risultato del server e valuta  $(C_{SO})$

Quale delle due strategie è meglio?

Se il server conosce o può fare inferenza sulle query il modello Client-server perde informazioni: il server può fare inferenza sulle tuple che sono associate con i valori che soddisfano  $(C_O)$ .

Se il server non conosce e non può fare inferenza sulle query: possiamo adottare qualsiasi strategia senza preoccuparci di violazioni di privacy (scegliendo magari quella più performante).

Il mio obiettivo è quello di minimizzare il carico di lavoro dell'owner.

La funzione peso  $w$  prende una coppia  $\langle F_O, F_S \rangle$  come input e ritorna il carico di lavoro dell'owner.

Una frammentazione  $F = \langle F_O, F_S \rangle$  è minimale sse:

1.  $F$  è corretta (soddisfa completezza, confidenzialità e non ridondanza)
2. non esiste  $F'$  tale che  $w(F') < w(F)$  e  $F'$  è corretta

Possiamo adottare diverse tecniche per dividere gli attributi tra  $F_O$  e  $F_S$  che minimizzano:

- storage: numero di attributi (Min-Attr) o dimensioni degli attributi (Min-Size)
- traffico/computazione: numero di query nelle quali è richiesto l'intervento dell'owner (Min-Query) o numero di condizioni all'interno delle query nelle quali l'owner deve essere coinvolto (Min-Cond)

Vediamole nel dettaglio:

- Min-Attr: minimizzo il numero di attributi nel frammento dell'owner
- Min-Size: minimizzo la dimensione fisica del frammento dell'owner (somma del peso fisico degli attributi)
- Min-Query: minimizzo il numero di query che richiedono lavoro all'owner (somma delle frequenze delle query)
- Min-Cond: minimizzo il numero di condizioni che richiedono del lavoro all'owner

Abbiamo così espresso tutte le metriche con un peso, tra queste cambia la forma del peso. A questo punto, dato il nostro problema e dato il nostro peso da minimizzare che ci verrà dato, dobbiamo trovare alcuni attributi da mettere in  $F_O$  e in  $F_S$  in modo da minimizzare il peso (uno dei quattro casi visti qui sopra). Rappresentiamo tutti i criteri con un modello uniforme basato su:

- insieme target: elementi rispetto ai quali è definito il problema di minimizzazione
- funzione peso: associa il peso a ogni elemento target
- peso dell'insieme di attributi: somma dei pesi dei target intersecati all'insieme

L'obiettivo è calcolare l'insieme di attributi con il minor peso, per farlo devo minimizzare la somma dei target che fanno parte dell'hitting set (uso un'euristica che mi assicura minimalità).

Questa euristica prende in input:

- A: insieme di attributi che non appaiono in vincoli singleton
- C: insieme di vincoli ben definiti
- T: insieme di target
- w: funzione peso definita su T

e come output restituisce:

- H: insieme di composizione di attributi insieme a quelli che appaiono con vincoli singleton
- $F_S$  è calcolato come  $R_O$  ottenendo una frammentazione corretta

Come struttura dati utilizziamo una Priority-queue PQ con un elemento E per ogni attributo:

- E.A: attributo
- E.C: puntatori a vincoli non soddisfatti che contengono E.A
- E.T: puntatori ai target che non intersecano H che contengono E.A
- E. $n_c$ : numero di vincoli puntati da E.C

- E.w: peso totale dei target puntati da E.T

La priorità è dettata da  $\frac{E.w}{E.n_c}$ : gli elementi con ratio minore hanno priorità più alta. Quello che vorrei fare è avere il peso minore, portandomi a casa il numero maggiore di constraint risolti.

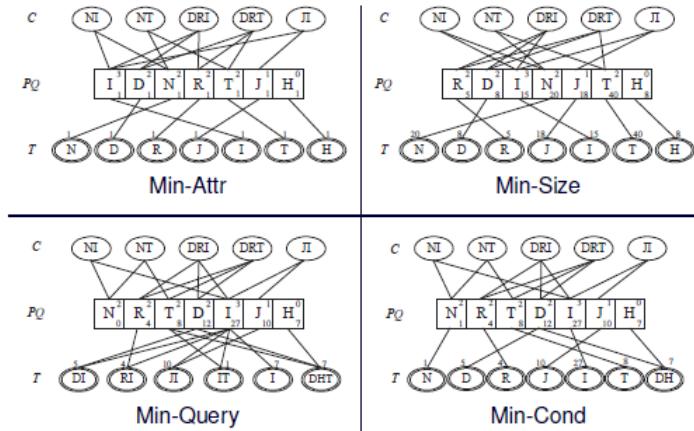
PATIENT(SSN,Name,DoB,Race,Job,Illness,Treatment,HDate)

**Confidentiality constraints**

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

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)



Quindi provando a esprimere questa euristica come un algoritmo:

- **WHILE** la coda di priorità PQ ha elementi ed esiste un elemento con numero di constraint non vuoto:
  - estrai l'elemento E con minore priorità data da  $\frac{E.w}{E.n_c}$
  - inserisci E.A in H
  - per ogni vincolo c puntato dall'elemento E.C, rimuovi il puntatore da c ad ogni altro elemento E' nella coda e aggiorna  $E'.n_c$
  - per ogni target t puntato dall'elemento E.T, rimuovi il puntatore da t ad ogni altro elemento E' nella coda e aggiorna  $E'.w$

- ricalcola la coda di priorità basata sui nuovi valori di priorità
- **FOREACH**  $A \in H$

- $\frac{H}{\{A\}}$  è un hitting set per C, rimuovi A da H

### 5.9.1 Frammentazione e inferenza

La frammentazione assume che gli attributi siano indipendenti. In presenza di dipendenza di dati abbiamo che gli attributi o le associazioni sensibili possono essere esposte indirettamente e allo stesso modo i frammenti possono essere indirettamente linkati.

## 5.10 Publishing Obfuscated Associations

Ci sono delle situazioni in cui voglio proteggere le associazioni sensibili (eventuali correlazioni), ma vorrei dare la possibilità di fare una certa query fintanto che non espongo un individuo. Ad esempio: numero medio di prodotti venduti da una farmacia senza rilevare il compratore.

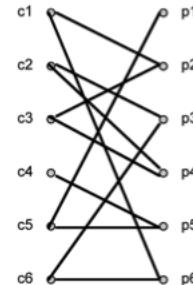
Andremo ora a vedere delle possibili soluzioni a questo problema.

### 5.10.1 Anonymizing Bipartite Graph

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



Customer e Product le possiamo rilasciare mentre Customer-Product no, perché è sensibile. Tuttavia vogliamo dare la possibilità di rispondere a certe query:

- **Tipo 0 - Solo struttura del grafo** ad esempio il numero medio di prodotti comprati
- **Tipo 1 - Attributi che hanno condizioni solo da una parte del grafo** ad esempio il numero medio dei prodotti comprati da persone che vivono a Milano

- **Tipo 2 - Attributi che hanno condizioni da entrambe le parti del grafo** ad esempio il numero medio di prodotti X comprati da persone che vivono a Milano

**(k,l) grouping** L'idea è quella di preservare la struttura del grafo ma permutare il mapping dalle entità ai nodi.

**(k,l) fa un raggruppamento di un grafo bipartito  $G = (V,W,E)$**

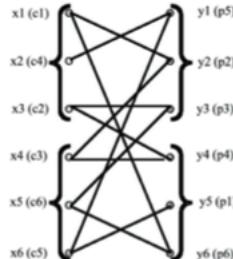
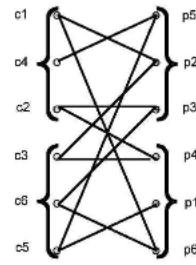
In ciascuna delle parti del grafo bipartito creo dei gruppetti, creo degli insiemi che non si intersecano (cioè una partizione) di dimensione almeno k.

Prendiamo come esempio un raggruppamento (3,3) a partire dall'esempio precedente:

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



x1   y2	Customer	Group	Product	Group	X-node	Group	Y-node	Group
x1   y6	c1	CG1	p1	PG2	x1	CG1	y1	PG1
x2   y1	c2	CG1	p2	PG1	x2	CG1	y2	PG1
x3   y3	c3	CG2	p3	PG1	x3	CG1	y3	PG1
x3   y4	c4	CG1	p4	PG2	x4	CG2	y4	PG2
x4   y2	c5	CG2	p5	PG1	x5	CG2	y5	PG2
x4   y4	c6	CG2	p6	PG2	x6	CG2	y6	PG2

$E'$        $H_V$        $H_W$        $R_V$        $R_W$

Ci sono metodi diversi per creare i raggruppamenti, ma non tutti i gruppi offrono lo stesso livello di privacy. Quello che cerchiamo di fare noi è il safe groupings: i nodi nello stesso gruppo di V non sono connessi ad uno stesso nodo in W.

La computazione di un raggruppamento safe può essere difficile anche per valori di k ed l piccoli.

Gli autori propongono un algoritmo greedy che iterativamente aggiunge un nodo a un gruppo con meno di k nodi se è safe.

Questo algoritmo funziona quando i grafi bipartiti sono abbastanza sparsi.

### 5.10.2 Fragments and Loose Associations

Questo metodo lavora ancora prima della frammentazione e ha come scopo quello di accrescere l'utilità del rilascio. Si parte sempre sapendo quali attributi (singleton) e quali associazioni sono sensibili.

Vengono introdotti dei requisiti di visibilità: formule booleane monotone sugli attributi rappresentano viste sui dati (le negazioni sono gestite dai vincoli di confidenzialità), Tali requisiti permettono di esprimere differenti livelli (di requisiti) di visibilità:

- attributi: alcuni attributi possono essere visibili
- associazioni: l'associazione tra valore di attributi dati possono essere visibili
- alternative views: almeno una delle viste specificate può essere visibile.

Ad esempio possiamo avere dei requisiti di visibilità sulla tabella seguente

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

che ci esplicitano cosa deve essere visibile. Ad esempio:

- $Patient \vee City$
- $(Birth \wedge City) \vee SSN$
- $Illness \wedge Doctor$

Quindi bisogna produrre delle viste che soddisfino i bisogni del cliente ma che non compromettano i vincoli di confidenzialità e che soddisfino i visibility constraints.

SSN	Patient	Birth	City	Illness	Doctor	
123-45-6789	Page	56/12/9	Rome	diabetes	David	$C_0 = \{SSN\}$
987-65-4321	Patrick	53/3/19	Paris	gastritis	Daisy	$C_1 = \{Patient, Illness\}$
963-85-2741	Patty	58/5/18	Oslo	flu	Damian	$C_2 = \{Patient, Doctor\}$
147-85-2369	Paul	53/12/9	Oslo	asthma	Daniel	$C_3 = \{Birth, City, Illness\}$
782-90-5280	Pearl	56/12/9	Rome	gastritis	Dorothy	$C_4 = \{Birth, City, Doctor\}$
816-52-7272	Philip	57/6/25	Paris	obesity	Drew	$V_1 = Patient \vee City$
872-62-5178	Phoebe	53/12/1	NY	measles	Dennis	$V_2 = (Birth \wedge City) \vee SSN$
712-81-7618	Piers	60/7/25	Rome	diabetes	Daisy	$V_3 = Illness \wedge Doctor$

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

La frammentazione è corretta se non viola i confidentiality constraints, soddisfa i vincoli di visibilità (se chiedo di vedere le stesse cose con un AND allora le devo mettere nello stesso frammento; con un OR me ne basta una), non mi espone a correlazioni (e infatti i frammenti non hanno attributi in comune - devo anche stare attento alle dipendenze).

Una frammentazione corretta è minimale se il numero di frammenti è minimo (cioè ogni altra frammentazione ha un numero uguale o maggiore di frammenti).

Il problema Min-CF, che riguarda il calcolare una frammentazione corretta e minima, è NP-hard.

Un SAT-Solver può risolvere in maniera efficiente il problema Min-CF. Un'istanza del problema Min-CF è tradotta in un problema SAT. Gli input del Min-CF sono visti come formule binarie.

Si itera la valutazione del SAT solver, partendo da un frammento e aumentando i frammenti di uno ad ogni iterazione, fino a quando non viene trovata una soluzione (che è sicuro che sia minimale).

La frammentazione ovviamente rompe le associazioni tra gli attributi. Per aumentare l'utilità delle informazioni pubblicate, i frammenti possono essere accoppiati con qualche associazione in forma sanificata (deve essere garantito un grado di privacy delle associazioni).

Per questo vengono utilizzate **loose associations** ossia delle associazioni tra gruppi di valori.

Nella pratica: Ho un frammento  $F_1$  e un frammento  $F_2$ . La loose association tra  $F_1$  e  $F_2$ :

- partiziona le tuple dei frammenti in gruppi
- fornisce informazioni sulle associazioni a livello di gruppo
- non permette la ricostruzione dell'associazione originale tra le tuple nel frammento
- fornisce un'utilità arricchita dei dati pubblicati

Quindi si pubblica a livello di gruppo (invece nel metodo Anonymizing Bipartite Graph si faceva aliasing).

Come funziona il k-grouping: vado a raggruppare le tuple in gruppetti che siano grandi almeno k. Un k-grouping fatto su un certo frammento è minimale se massimizza il numero di gruppi. Raggruppo in  $k_1$  e  $k_2$ . Un raggruppamento  $(k_1, k_1)$  è minimale se da entrambe le parti si è fatto un raggruppamento minimale.

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

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

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

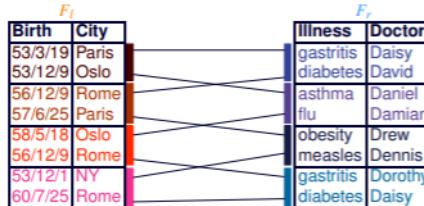
Un raggruppamento ( $k_l, k_r$ ) introduce un'associazione di gruppo A tra i gruppi in  $f_l$  e  $f_r$ .

Un'associazione di gruppo A tra i due frammenti  $f_l$  e  $f_r$  è una coppia di gruppi di identificatori tali che:

- A ha la stessa cardinalità della relazione originale
- c'è un mapping biettivo tra la relazione originale e A che associa ogni tupla nella relazione originale con una coppia  $(G_l(l), G_r(r))$  in A, con  $l \in f_l$  e  $r \in f_r$

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

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



Pubblichiamo le associazioni a livello di gruppo e non a livello della singola tupla e il risultato è il seguente:

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

$C_0 = \{\text{SSN}\}$   
 $C_1 = \{\text{Patient}, \text{Illness}\}$   
 $C_2 = \{\text{Patient}, \text{Doctor}\}$   
 $C_3 = \{\text{Birth}, \text{City}, \text{Illness}\}$   
 $C_4 = \{\text{Birth}, \text{City}, \text{Doctor}\}$

$F_l$		$F_r$	
Birth	City	G	G
53/3/19	Paris	bc1	bc1 id1
53/12/9	Oslo	bc1	bc1 id2
56/12/9	Rome	bc2	bc2 id1
57/6/25	Paris	bc2	bc2 id3
58/5/18	Oslo	bc3	bc3 id2
56/12/9	Rome	bc3	bc3 id4
53/12/1	NY	bc4	bc4 id3
60/7/25	Rome	bc4	bc4 id4

La pubblicazione dei gruppi è fatta in modo loose, quindi aumento l'incertezza di  $2 \times 2 = 4$  (ad ogni elemento potrebbero corrispondere 4).

I duplicati nei frammenti vengono mantenuti (siccome tutti i frammenti hanno la stessa cardinalità della relazione originale) e quindi potrebbero contenere tuple uguali.

Anche delle tuple diverse potrebbero avere stessi valori per gli attributi coinvolti nei vincoli di confidenzialità. La protezione looseness offerta dal raggruppamento potrebbe quindi essere compromessa: dobbiamo controllare le occorenze degli stessi valori.

Due tuple sono simili (alike  $\simeq$ ) rispetto ad un vincolo c se:

- il vincolo c è coperto dall'unione (inteso come unione di attributi) dei due frammenti  $F_r$  e  $F_l$
- una tupla che si interseca con i vincoli è uguale all'altra tupla che si interseca con i vincoli

Questa proprietà di somiglianza è transitiva per ciascun vincolo c.

Non è transitiva invece se ci sono almeno due vincoli coperti da  $F_r$  e  $F_l$

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

$C_0 = \{\text{SSN}\}$   
 $C_1 = \{\text{Patient}, \text{Illness}\}$   
 $C_2 = \{\text{Patient}, \text{Doctor}\}$   
 $C_3 = \{\text{Birth}, \text{City}, \text{Illness}\}$   
 $C_4 = \{\text{Birth}, \text{City}, \text{Doctor}\}$

$F_l$		$F_r$	
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

$\simeq_{C_4}$   
 $\simeq_{C_3}$

Un'associazione a livello di gruppo è k-loose se ogni tupla in un gruppo può corrispondere dall'altra parte ad almeno k associazioni diverse. Proprietà: se sei k-loose su 4 allora sei k-loose anche su 3, su 2, ...

Un'associazione di gruppo A è minimale su  $(k_l, k_r)$  se:

- A è k-loose
- non era possibile fare di meglio (cioè non c'era un'altra associazione con valori più piccoli che dava lo stesso grado di loose).

Se vogliamo garantirci un'associazione k-loose allora tale associazione a livello di gruppo deve tocare tuple che sono diverse e non sono simili.

Se un raggruppamento  $(k_l, k_r)$  soddisfa le seguenti proprietà di eterogeneità:

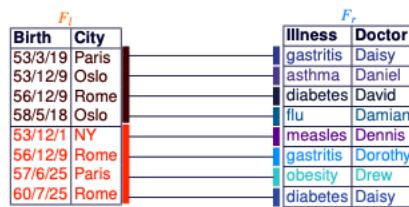
- di gruppo: nessun gruppo può contenere tuple che sono simili rispetto a qualche vincolo
- di associazione: nessun gruppo può essere associato due volte con lo stesso gruppo, altrimenti perderei l'incertezza
- profonda: nessun gruppo può essere associato con due gruppi che hanno tuple simili

allora l'associazione tra i gruppi è k-loose con  $k=k_l \times k_r$ .

Un raggruppamento  $(k_l, k_r)$  è:

- flat se almeno uno tra  $k_l$  e  $k_r$  sono uguali a 1. In questo caso assomiglia a k-anonymity e l-diversity insieme, ma lavora sulle associazioni e i valori degli attributi non sono generalizzati

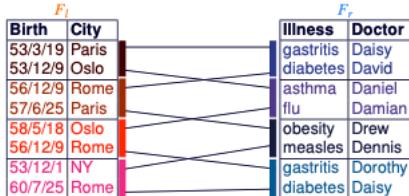
Birth	City	Illness	Doctor	
56/12/9	Rome	diabetes	David	$C_0 = \{\text{SSN}\}$
53/3/19	Paris	gastritis	Daisy	$C_1 = \{\text{Patient}, \text{Illness}\}$
58/5/18	Oslo	flu	Damian	$C_2 = \{\text{Patient}, \text{Doctor}\}$
53/12/9	Oslo	asthma	Daniel	$C_3 = \{\text{Birth}, \text{City}, \text{Illness}\}$
56/12/9	Rome	gastritis	Dorothy	$C_4 = \{\text{Birth}, \text{City}, \text{Doctor}\}$
57/6/25	Paris	obesity	Drew	
53/12/1	NY	measles	Dennis	
60/7/25	Rome	diabetes	Daisy	



- sparse se sia  $k_l$  che  $k_r$  sono diversi da 1. Garantisce una larga applicabilità rispetto al raggruppamento flat, con lo stesso livello di protezione

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

$C_0 = \{\text{SSN}\}$   
 $C_1 = \{\text{Patient}, \text{Illness}\}$   
 $C_2 = \{\text{Patient}, \text{Doctor}\}$   
 $C_3 = \{\text{Birth}, \text{City}, \text{Illness}\}$   
 $C_4 = \{\text{Birth}, \text{City}, \text{Doctor}\}$



La pubblicazione di associazioni loose incrementa l'utilità dei dati poichè rende possibile la valutazione di query in modo più preciso rispetto alla pubblicazione dei soli frammenti. Questo però comporta ad un'esposizione delle informazioni (grado di privacy più basso).

L'esposizione di un'associazione sensibile  $\langle l[c \cap F_l], r[c \cap F_r] \rangle$  con  $c$  che è un vincolo coperto da  $F_l$  e  $F_r$  può essere espressa come la probabilità che ci sia quell'associazione nella relazione originale.

L'incremento di esposizione derivata dalla pubblicazione di un'associazione loose può essere misurata come la differenza tra:

- la probabilità  $P^A(l[c \cap F_l], r[c \cap F_r])$  che l'associazione sensibile  $\langle l[c \cap F_l], r[c \cap F_r] \rangle$  compaia nella relazione originale dati  $f_l, f_r$  e A
- la probabilità  $P(l[c \cap F_l], r[c \cap F_r])$  che l'associazione sensibile  $\langle l[c \cap F_l], r[c \cap F_r] \rangle$  compaia nella relazione originale dati  $f_l$  e  $f_r$

Dati  $l \in f_l$  e  $r \in f_r$  la probabilità  $P(l, r)$  che la tupla  $\langle l, r \rangle$  appartenga alla relazione originale è  $\frac{1}{|f_l|} = \frac{1}{|f_r|}$

	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
53/12/9	Oslo	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
57/6/25	Paris	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
56/12/9	Rome	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
60/7/25	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8

L'esposizione di  $(P(l[c \cap F_l], r[c \cap F_r]))$  dipende dalla presenza di tuple simili. Siano  $l_i, l_j$  due tuple in  $f_l$  tali che  $l_i \simeq_c l_j$ ,  $P(l_i[c \cap F_l], r[c \cap F_r])$  è la composizione delle probabilità che:

- $l_i$  sia associato con r
- $l_j$  sia associato con r

$$P(l_i, r) + P(l_j, r) - (P(l_i, r) * P(l_j, r))$$

Dalla tabella precedente (considerando il vincolo 3 = Birth, City, Illness) dobbiamo togliere le tuple simili che mi fanno diventare la tabella come segue (abbiamo usato la formula definita precedentemente di combinazione di probabilità):

		gastritis	diabetes	asthma	flu	obesity	measles
		15/64	15/64	1/8	1/8	1/8	1/8
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

Questo si verifica nel caso in cui si decidesse di non pubblicare le associazioni loose. Nel caso contrario invece si partirebbe da una situazione simile a quella seguente:

	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
56/12/9	Rome	—	—	1/4	1/4	—	—	1/4
53/12/1	NY	—	—	—	—	1/4	1/4	1/4
60/7/25	Rome	—	—	—	—	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

Dati  $l \in f_l$  e  $r \in f_r$  la probabilità  $P^A(l, r)$  che la tupla  $\langle l, r \rangle$  appartiene alla relazione originale è almeno  $\frac{1}{k}$ .

$P^A(l[c \cap F_l], r[c \cap F_r])$  è valutata considerando la somiglianza. Siano  $l_i, l_j$  due tuple in  $f_l$  tali che  $l_i \simeq_c l_j$ ,  $P^A(l_i[c \cap F_l], r[c \cap F_r])$  è la composizione delle probabilità che:

- $l_i$  sia associato con  $r$
- $l_j$  sia associato con  $r$

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

Applichiamo lo stesso procedimento visto in precedenza e otterremo:

		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

Abbiamo parlato di utilità e privacy dei dati. Come possiamo misurarle?

- **utilità:** media delle differenze di probabilità delle due tabelle (con associazioni e senza associazioni) per ogni associazione sensibile
- **privacy:** data una soglia di privacy verifico che la differenza sopra citata non vada mai a superarla

## 6 Confidenzialità e integrità delle query

Garantire la privacy dei dati pubblicati protegge la confidenzialità dei dati (confidenzialità del contenuto) come anche l'accesso agli stessi:

- **confidenzialità di accesso:** riservatezza del fatto che un accesso mira a un dato specifico
- **confidenzialità di pattern:** riservatezza del fatto che due accessi mirano agli stessi dati

Esistono diversi approcci per proteggere l'accesso ai dati e nelle prossime sezioni andremo ad approfondirne alcuni.

### 6.1 Path ORAM

L'idea che sta alla base è quella di eseguire query senza dire al server cosa sto cercando.

Lato **server**: i dati sono organizzati in un albero binario, dove ciascun nodo è un blocco dove metto dentro più cose (dati reali e spazio, dummy blocks) e dove c'è un unico cammino che va dalla foglia alla radice.

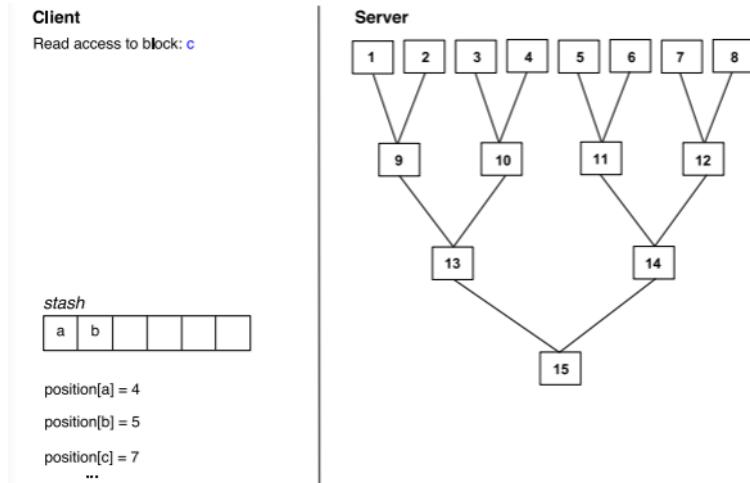
Lato **client**: si tiene un po' di memoria localmente (stash), si tiene una mappa delle chiavi (in che posizione è mappata la chiave). Se il blocco esiste, sta da qualche parte nel cammino dalla radice alla posizione della chiave. La mappa di posizionamento cambia ogni volta che si accede o si rimappa un blocco.

In ogni momento ogni blocco è mappato uniformemente e in maniera randomica in un bucket delle foglie e blocchi unstashed sono posizionati sempre in qualche bucket lungo il percorso che va alla foglia mappata.

Le operazioni che possiamo fare sono:

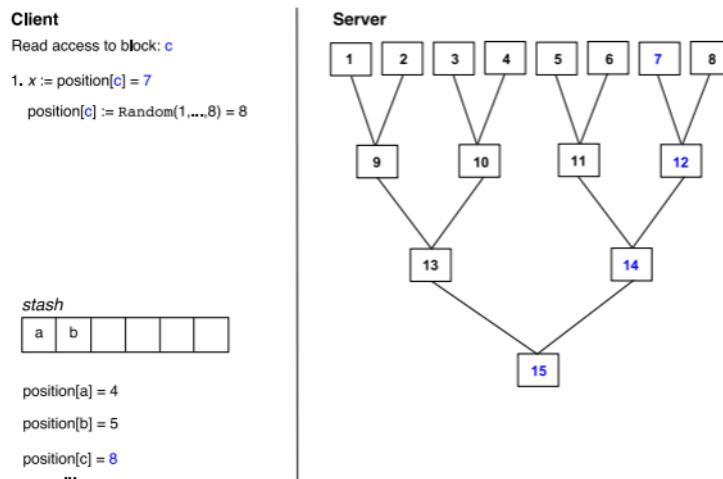
1. rimappare i blocchi: sia x la vecchia posizione di a. Si rimappa randomicamente la posizione di a in una nuova posizione (un nuovo nodo foglia)
2. leggere il percorso: legge i nodi che in  $P(x)$  contengono a. Se l'accesso è una scrittura si aggiorna il dato salvato nel blocco a.
3. scrivere il percorso: scrive i nodi in  $P(x)$  includendo qualche blocco aggiuntivo dallo stash se possono essere inclusi nel percorso (deve essere verificata l'invariante principale)

Vediamo un esempio:

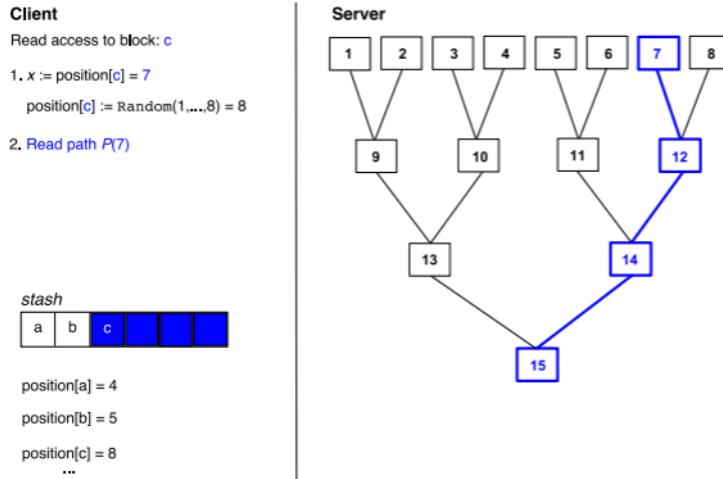


La posizione di a è 4: significa che a si trova da qualche parte lungo il cammino dalla radice al bucket 4. Il client si tiene il dizionario per tutte le chiavi. Lo stash è come se fosse una cache locale. Ogni volta che qualcuno legge una cosa, cambi il posto. Così se un altro utente andrà a leggere la stessa cosa, avrà un percorso diverso. Come funziona l'accesso?

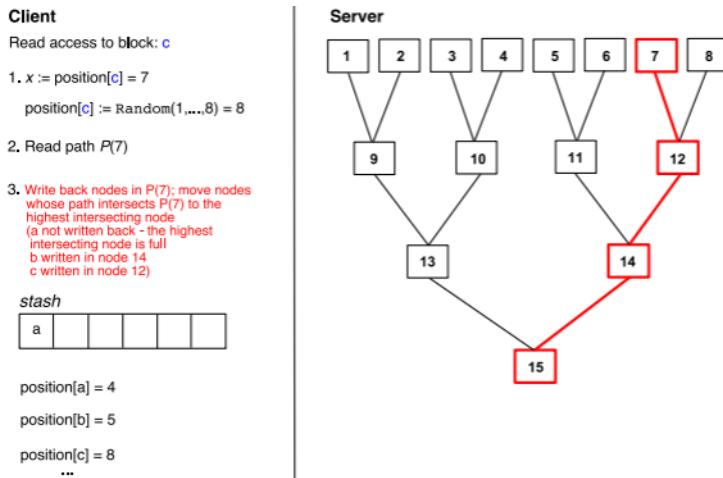
L'utente vuole leggere il valore di una certa chiave



Leggo il cammino a cui trovo c



Leggendo scarico il più possibile la mia stash (elenco di cose che ho letto ma che devo ancora riscrivere). Cioè già che faccio quel percorso, mappo nel percorso le lettere che sono ancora nella stash.



**Problema di Path ORAM:** ogni lettura, che solitamente non è concorrente, diventa anche scrittura, che invece è concorrente. Quindi diventa pesante, soprattutto se ci sono tanti accessi multipli.

## 6.2 Ring ORAM

Variazione di Path ORAM che riduce l'accesso alla banda online a O(1). All'interno dei nodi ho: dei blocchi addizionali, una piccola mappa degli offsets dei blocchi, un contatore degli accessi.

La lettura è a livello dei singoli blocchetti, quindi vengono letti anche i blocchetti vuoti. Il contatore serve per fare in modo che la lettura di questi blocchetti vuoti non venga scoperta. Non scrive tutte le volte che va a leggere ma solo quando i contatori dicono che le letture potrebbero diventare osservabili.

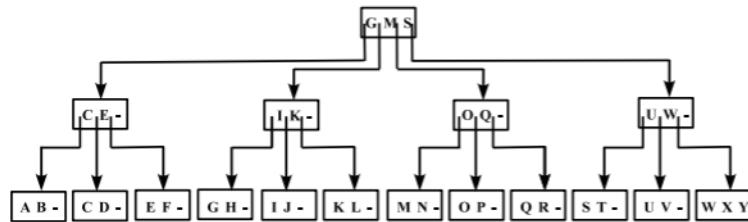
### 6.3 Shuffle Index

La questione di base è sempre quella: se voglio proteggere la privacy devo rompere il legame tra il dato e la posizione del dato.

Lato **client**: organizzo le chiavi come un albero di ricerca, non un albero binario ma un albero paginato (B+). Però in questo caso le foglie non sono collegate (chained) ma sono scollegate (unchained) perché così le posso mischiare (shuffle). Anche perché il puntatore delle foglie può fare leakage del fatto che una foglia viene prima di un'altra.

I nodi hanno la seguente struttura:

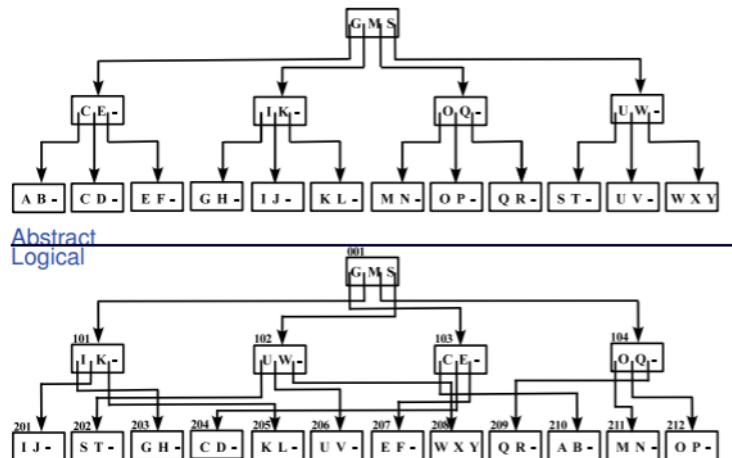
- ciascun nodo deve essere riempito almeno della metà della sua capienza, così da riempire bene l'albero
- l'i-esimo figlio è la radice del sottoalbero che contiene i nodi v tali per cui i figli stanno tra le due chiavi ( $\geq$ )



#### 6.3.1 Rappresentazione a livello logico

I puntatori tra i nodi della struttura astratta corrispondono (a livello logico) agli identificatori dei nodi.

Sono rappresentati da un insieme di coppie  $(id, n)$  con id che è l'identificatore e n il contenuto del nodo: l'ordine tra gli identificatori non corrisponde necessariamente all'ordine in cui i nodi appaiono nella rappresentazione astratta

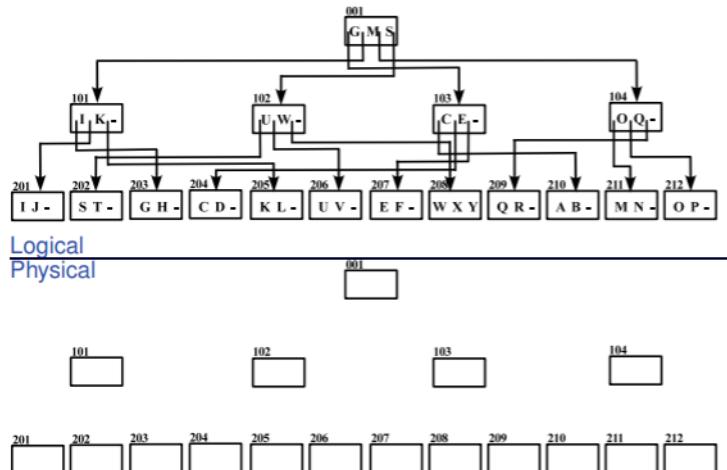


### 6.3.2 Rappresentazione a livello fisico

Ogni nodo  $\langle id, n \rangle$  nello shuffle index a livello logico è salvato sul server in maniera criptata.

Ogni nodo  $\langle id, n \rangle$  corrisponde a un blocco  $\langle id, b \rangle$  dove b è un blob che contiene il contenuto criptato con la sua firma e un sale random così che ogni criptazione sia diversa. Alla fine un MAC con chiave critta il contenuto con l'identificatore, perché non posso spostare a caso il contenuto dei nodi.

Quindi: critto il contenuto delle foglie, lo concateno con il suo id e faccio un hash con chiave per firmare (id, b).



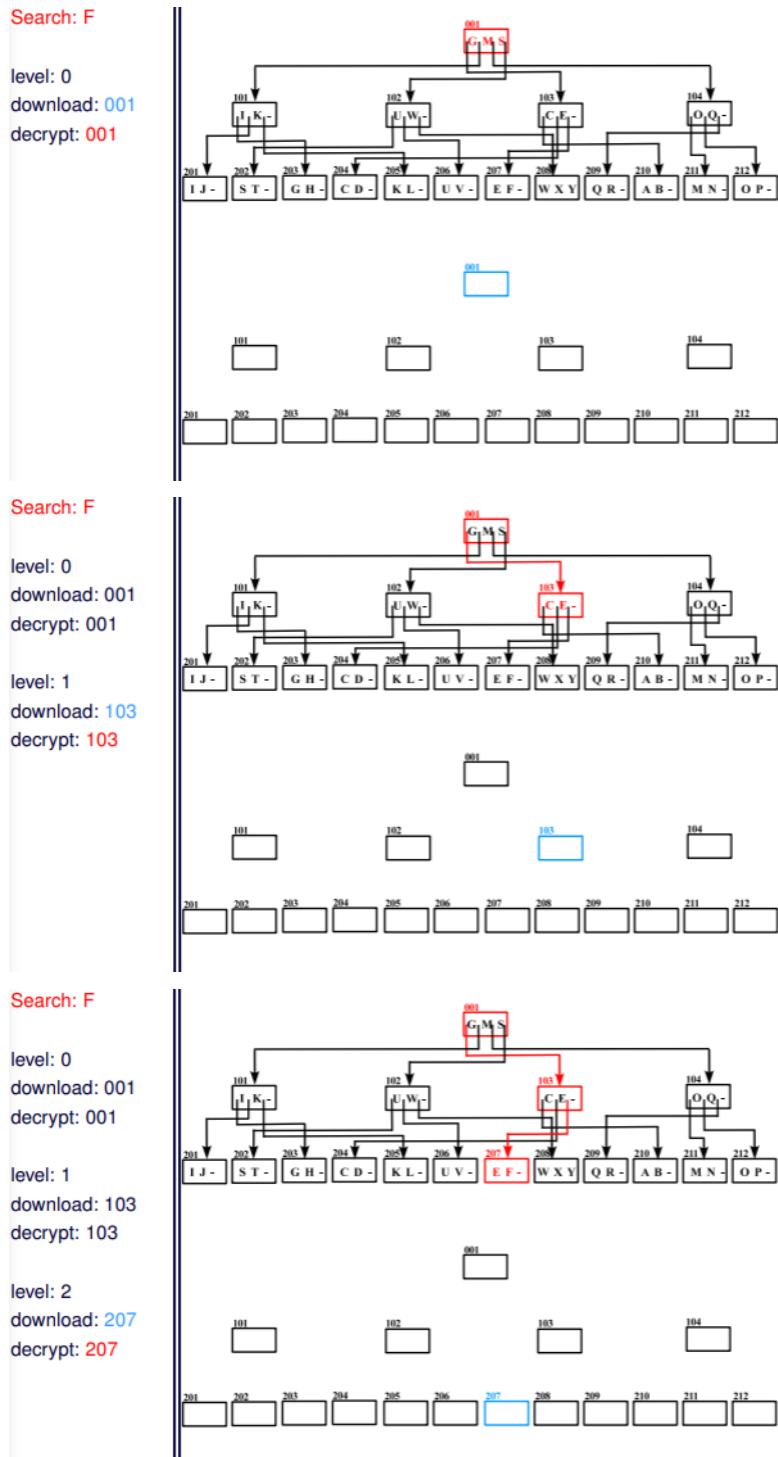
Avendo tutto criptato, non è il server che mi dice quale puntatore seguire. Dovrà farlo il client decriptato il nodo.

### 6.3.3 Accesso ai dati

Accedere ai dati richiede un processo iterativo tra client e server. Il client esegue un'iterazione per ogni livello di shuffle index partendo dalla radice. Ad ogni iterazione il client:

- decripta il blocco ricevuto
- determina il blocco che deve essere recuperato dal server al prossimo livello

Il processo termina quando viene recuperato un blocco foglia.



Il server riceve un insieme di blocchi da salvare e le richieste di accedere ai blocchi che vengono tradotte in osservazioni: un'osservazione  $o_i$  corrisponde a una sequenza

di blocchi.

Il server può quindi fare molto facilmente inferenze su:

- il numero  $m$  di blocchi e i loro identificatori
- l'altezza  $h$  del shuffle index
- il livello associato ad ogni blocco

Data una sequenza di osservazioni il server non dovrebbe essere capace di effettuare inferenze su:

- dati salvati nel shuffle index (confidenzialità del contenuto)
- l'accesso ai dati si riferisce ad un nodo specifico (confidenzialità di accesso)
- $o_i$  ha lo scopo di accedere allo stesso nodo di  $o_j$  (confidenzialità di pattern)

La criptazione lato server protegge la confidenzialità del contenuto e la confidenzialità dell'accesso per ogni richiesta individuale.

Tuttavia non vengono protetti confidenzialità di accesso e pattern dato che accedere agli stessi blocchi implica accedere agli stessi dati. Si possono eseguire attacchi di frequenza che permettono di ricostruire la corrispondenza tra il valore degli attributi in chiaro e i blocchi.

Per distruggere questa corrispondenza si combinano tre strategie:

1. **cover searches**: aggiungo rumore (mi interessa una chiave ma ne aggiungo altre alla ricerca)
2. **cached searches**: tengo qualche informazione lato client
3. **shuffling**: cambia dinamicamente l'allocazione dei blocchi ad ogni accesso

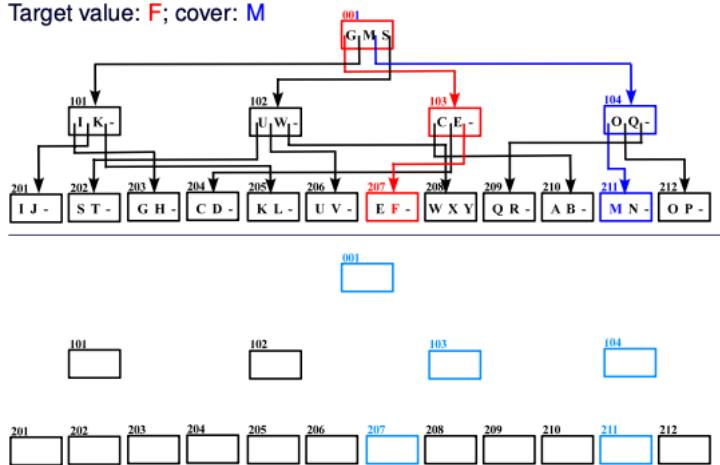
#### 6.3.4 Cover searches

Introduco confusione aggiungendo richieste di cui in realtà non mi importa.

Il numero di coperture (num\_cover) è un parametro di protezione.

Questa strategia deve:

- fornire una diversità tra i blocchi (differenza tra i percorsi dei blocchi che mi interessano e i blocchi che introducono rumore)
- essere indistinguibile dalla ricerca effettiva



Potremmo subire attacchi di inserzione: la query vera rimane sempre uguale mentre quella di copertura cambia ogni volta.

### 6.3.5 Cached searches

Il client mantiene una cache locale con gli ultimi accessi effettuati.

Se c'è un nodo in cache allora anche suo padre deve stare in cache (path continuity). La cache cambia ad ogni accesso.

Se un nodo è nella cache posso o non fare nulla o fare ricerca cover.

Questo metodo protegge dagli attacchi di inserzione a breve termine, mentre può essere soggetto ad attacchi di inserzione che vanno oltre la dimensione della cache.

### 6.3.6 Shuffling

Questa tecnica va a rompere la corrispondenza uno a uno tra blocchi e nodi, cambiando il contenuto tra i nodi.

Il shuffling richiede decryption e re-encryption: ad ogni spostamento il nodo va recriptato con un sale diverso. Se cambio i nodi devo anche cambiare il puntatore verso i genitori.

Prendendo in considerazione tutte e tre le tecniche abbiamo:

- degradazione dovuta allo shuffling
- confidenzialità di accesso: ogni volta che viene fatto l'accesso a un'informazione viene aggiunto un rumore che distrugge la corrispondenza nodo-blocco
- confidenzialità di pattern: a breve termine con cover e cache, a lungo termine con cover e shuffling

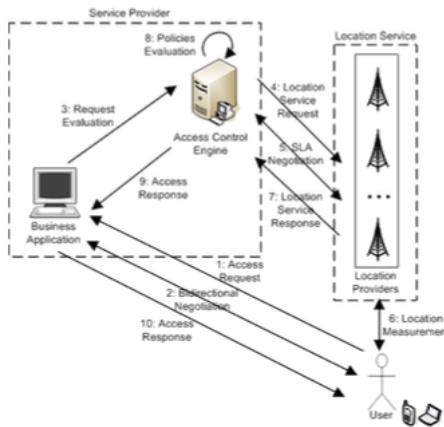
È sicuramente un metodo molto costoso: num\_cover + num\_cache + una scrittura sul server. Ha prestazioni migliori di Path ORAM

## 7 Privacy e dati di locazione

L'idea alla base è quella di utilizzare la geolocalizzazione per controllare l'accesso. Questo concept presenta diversi vantaggi:

- naturale per certi tipi di applicazioni
- può permettere di accedere alle risorse ai soli individui di un edificio

Migliora sicuramente il controlli dell'accesso con ulteriori capacità di specificare, valutare e rafforzare le condizioni basate sulla locazione.



Abbiamo due grandi attori: location provider che gestisce e analizza la posizione, service provider che offre il servizio che può usare la posizione.

Tutti i sistemi di geolocalizzazione hanno un margine di errore dovuto a limitazioni tecniche e effetti ambientali. La relevance è una soglia che misura l'accuratezza del dato di location: 0 se non è affidabile, 1 è perfetta e da 0 a 1 se è meno dell'ottimo a causa di errori di misurazione e degradazione artificiale.

Mi interessano due valori:

- livello di accuratezza della posizione ( $R_{Eval}$ )
- livello minimo di accuratezza che il service provider chiede ( $R_{LBAC}$ )

Le query al Location Service Provider LP possono essere:

- **Range queries**: prendono soltanto dei parametri e restituiscono range, accuratezza, timeout. Esempio: dove si trova Mario Rossi.
- **Boolean queries**: valutano un predicato che ha in input dei parametri e dei valori e restituiscono un booleano, la relavance e un timeout. Esempio: Mario Rossi è in questa posizione?

Assumiamo l'esistenza degli elementi seguenti:

- utenti: insieme degli identificatori UID che identificano in maniera non ambigua l'utente

- aree: insieme di regioni della mappa identificate tramite modelli geometrici o modelli simbolici

Esistono principalmente tre classi di condizioni:

- position-based: valutano la posizione dell'utente
- movement-based: valutano la mobilità dell'utente (velocità, accelerazione, direzione)
- interaction-based: relazionano più utenti e/o entità

Per quanto riguarda i predicati invece abbiamo:

- predici di posizione:
  - inarea(user, area): valuta se l'utente è all'interno di una certa area
  - disjoint(user, area): valuta se l'utente è fuori dall'area
  - distance(user, entity, min\_dist, max\_dist): valuta se la distanza tra l'utente e l'entità è tra una distanza minima e una massima
- predici di movimento:
  - velocity(user, min\_vel, max\_vel): valuta se la velocità dell'utente è tra un valore massimo e un valore minimo
- predici di interazione:
  - density(area, min\_num, max\_num): valuta se il numero di utenti presenti nell'area sono all'interno di un numero minimo e un numero massimo
  - local\_densitu(user, area, min\_num, max\_num): valuta la densità all'interno dell'area dove è presente un determinato utente

Ad esempio: inarea(Alice, Milan) = [True, 0.9, 2005-11-09-11:10am] vuol dire che è vero che Alice si trova a Milano con una confidenza del 90% e che questa informazione è da ritenersi vera fino alle 11:10am del 9 novembre 2005.

Le regole del linguaggio basato sulla posizione sono del tipo  $\langle subj\_expr, obj\_expr, action \rangle$  dove:

- subj\_expr: formule booleane di termini che specificano condizioni sul soggetto
- obj\_expr: formule booleane di termini che specificano condizioni su oggetti
- action: azione o classe di azioni alle quali si riferiscono le regole

I profili sono referenziati con l'identità dell'utente/oggetto corrispondente. Esistono tuttavia tre variabili predefinite che rendono possibile il riferimento delle condizioni delle regole al richiedente e all'oggetto:

- user: identificatore della persona che fa la richiesta
- sim: il numero di sim della persona che fa la richiesta
- object: identificatore dell'oggetto al quale è richiesto l'accesso

subject		
generic conditions		location conditions
1	<code>user.Role="Admin" <math>\wedge</math> Valid(user.Username, user.Password)</code>	<code>density ("Server Room", 1, 1) <math>\wedge</math> velocity (sim, 0, 3) <math>\wedge</math> inarea (sim, "Server Room")</code>
2	<code>user.Role="Admin" <math>\wedge</math> Valid(user.Username, user.Password)</code>	<code>local_density (sim, "CloseBy", 1, 1) <math>\wedge</math> velocity (sim, 0, 3) <math>\wedge</math> inarea (sim, "Inf. System Dept.")</code>
3	<code>user.Role="CEO" <math>\wedge</math> Valid(user.Username, user.Password)</code>	<code>local_density(sim, "CloseBy", 1, 1) <math>\wedge</math> velocity (sim, 0, 3) <math>\wedge</math> inarea (sim, "Corporate Main Office")</code>
4	<code>user.Role="CEO" <math>\wedge</math> Valid(user.Username, user.Password)</code>	<code>local_density (sim, "CloseBy", 1, 1) <math>\wedge</math> disjoint (sim, "Competitor Location")</code>
5	<code>user.Role="Guest" <math>\wedge</math> Valid(user.Username, user.Password)</code>	<code>local_density(sim, "CloseBy", 1, 1) <math>\wedge</math> inarea (sim, "Corporate Location")</code>

### 7.0.1 Valutazione di LBAC: da $R_{eval}$ a valori di verità

I predicati location-based sono formule booleane, mentre la risposta al predicato è nella forma  $[bool\_value, R_{Eval}, timeout]$ . Come facciamo ad ottenere un valore di verità? Possiamo fare uso di tabelle di verità estese che specificano il mapping tra predicati e valori di verità.

ACE specifica per ogni predicato e ogni servizio di localizzazione:

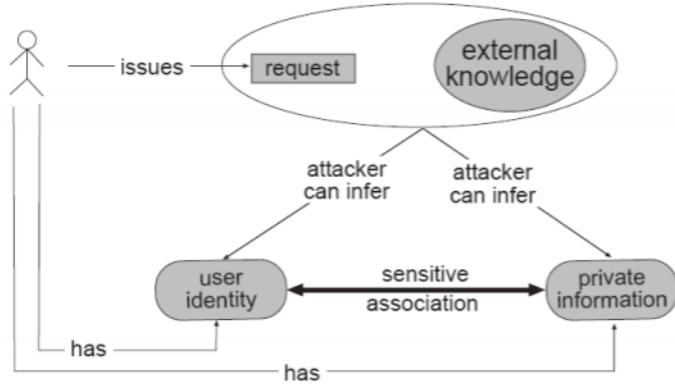
- rilevanza  $R_{LBAC}$  risultante da due valori soglia (lower, upper) =  $((1-R_{LBAC}), R_{LBAC})$ 
  - $R_{eval} \leq lower$ : !bool\_value è considerato valido
  - $R_{eval} \geq lower$ : bool\_value è considerato valido
  - $lower \leq R_{eval} \leq higher$ : la misura è ripetuta
- massimo numero di tentativi concessi (MaxTries)

Predicate	$\mathcal{R}_{LBAC}$ Thresholds		MaxTries
	lower	upper	
inarea	0.1	0.9	10
disjoint	0.1	0.9	10
distance	0.2	0.8	5
velocity	0.2	0.8	5
density	0.3	0.7	3
local density	0.3	0.7	3

LBAC può essere utilizzato come rinforzo, infatti ricevendo una richiesta di accesso nella forma  $\langle user_id, action, object \rangle$ :

1. seleziona le regole applicabili
2. valuta le condizioni generiche
3. se c'è una regola che è già soddisfatta: garantisci l'accesso
4. altrimenti valuta le condizioni LB:
  - true  $\Rightarrow$  garantisci l'accesso
  - false  $\Rightarrow$  nega l'accesso
  - unknown  $\Rightarrow$  nega l'accesso

## 7.1 Protezione delle informazioni di locazione

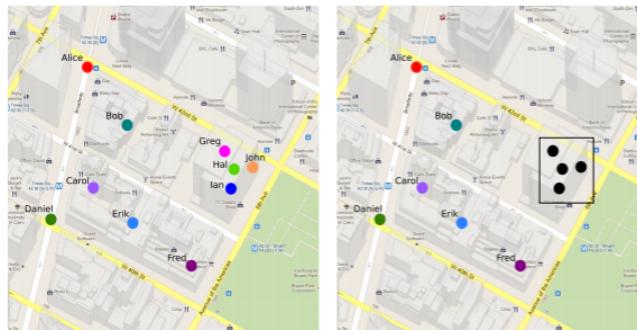


Location privacy: il diritto dell'utente di decidere come, quando e per quale scopo le loro informazioni di locazione possono essere rilasciate:

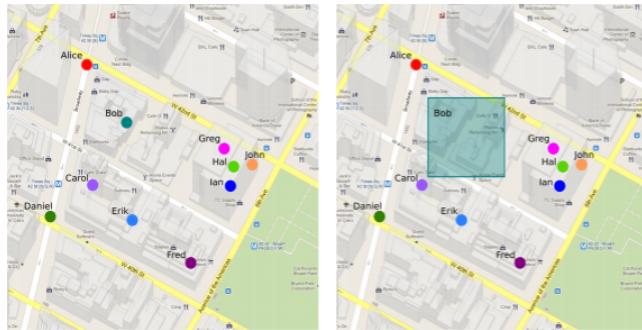
- privacy d'identità: protegge l'identità dell'utente associato all'informazione di locazione
- privacy di posizione: protegge le informazioni di locazione degli utenti perturbandole
- privacy di percorso: protegge il percorso degli utenti

Abbiamo a disposizione diversi tipi di soluzioni per proteggere le informazioni sulla posizione:

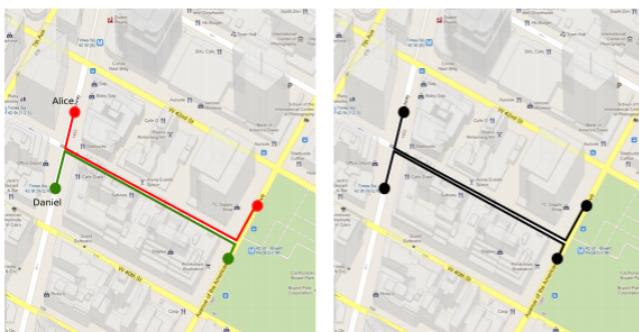
- tecniche anonymity-based: si basano sul concetto di anonimia. Definite in principio per proteggere la privacy di identità, sono meno adatte per proteggere la privacy di posizione, ma più adatte per proteggere la privacy di percorso



- tecniche obfuscation-based: si basano sul concetto di offuscazione. Definite in principio per proteggere la privacy di posizione, sono meno adatte per proteggere la privacy di identità, ma più adatte per proteggere la privacy di percorso



- tecniche policy-based: si basano sul concetto di privacy policy. Sono ottime per proteggere tutti i tipi di privacy citati sopra



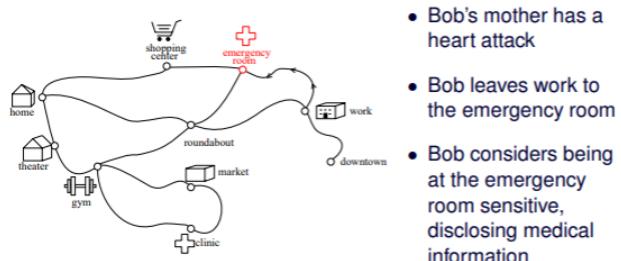
Infrastrutture mobile e basate sui servizi sono caratterizzate da utenti mobile e LBS providers dove:

- LBS providers offrono servizi richiedendo il continuo accesso alla posizione degli utenti
- utenti mobile rilasciano le loro posizioni ai LBS providers

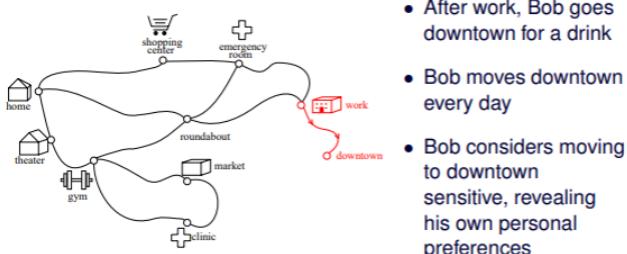
Un LBS provider che analizza le posizioni dell'utente potrebbe effettuare inferenze sulle informazioni sensibili dell'utente e violare la sua privacy.

Ci sono tre classi di inferenze possibili:

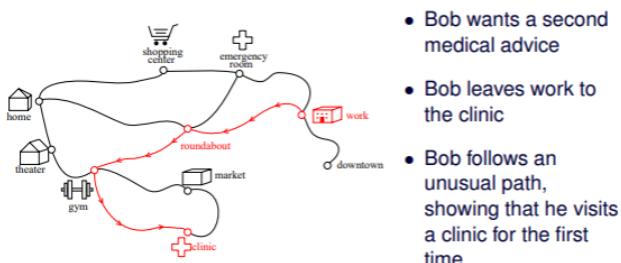
1. sensitive positions (l'utente ha visitato un posto che considera sensibile: user-dependent)



2. sensitive movements (l'utente ha seguito un percorso che ritiene sensibile: user-dependent)



3. unusual paths (un utente ha seguito un percorso inusuale per lui: user-independent)



La comunicazione tra utenti e providers è mediata da un trusted privacy middleware che:

1. valuta le informazioni di locazione degli utenti
2. calcola il rischio di inferenza
3. offusca il percorso prima di rilasciarlo se è possibile effettuare delle inferenze (cover stories basato sulle preferenze dell'utente)

La rete viene modellata come un grafo  $G(V,E)$  dove  $V$  sono le intersezioni tra strade e i punti di interesse ed  $E$  sono le strade.

L'utente, basandosi su questo grafo, può esprimere preferenze di privacy e servizio di qualità.