Privatezza e Protezione dei Dati - Papers

Francesco Fontana, Alessandro Marchetti, Alfredo Santarcangelo

2024

Contents

0.1 Introduction

Chapter 1

k-anonimity

Contesto è il rilascio di *microdata*. De-identificazione non garantisce anonimità.

1.1 *k*-anonimity e Table *k*-anonime

Il concetto di k-anonimity cerca di catturare sulla Private Table (PT) il vincolo che i dati rilasciati dovrebbero essere associabili in maniera indistinguibile a non meno di un certo numero di respondent.
Il set di attributi disponibili esternamente e quindi sfruttabili per fare linking è chiamato *quasi-identifier*.

Definizione 1 (*k*-anonimity requirement)
Ogni rilascio di data deve essere tale che ogni combinazione di valori del Quasi-Identifier può essere matchata in maniera indistinguibile con almeno k respondent.

k-anonimity richiede che ogni valore del *Quasi-Identifier*abbia almeno k occorrenze nella table rilasciata, come da def 1.1 che segue:

Definizione 2 (*k*-anonimity)
Date una table $T(A_1, A_2, ...A_m)$ *e un insieme di attributi* QI*, Quasi-Identifier sulla table* T*:*
T *soddisfa k-anonimity rispetto a* QI *se e solo se ogni sequenza di valori in* $T[QI]$ *appare almeno con k occorrenze in* $T[QI]$[1]

Definizione 1.1 è sufficiente per k-anonimity. Applicazione di k-anonimity richiede una preliminare identificazione del *Quasi-Identifier*.

Il *Quasi-Identifier* dipende dalle informazioni esterne disponibili al recipiente poichè determina le capacità di linking dello stesso. Diversi *Quasi-Identifier* possono potenzialmente esistere per una data table.

Per semplicità a seguire nel paper si assume che:

[1] $T[QI]$ denota la proiezione con tuple duplicate degli attributi QI in T

- PT ha unico *Quasi-Identifier*.
- *Quasi-Identifier* è composto da tutti gli attributi nella PT disponibili esternamente.
- PT contiene al massimo una sola tupla per ogni respondent.

k-anonimity si concentra su due tecniche di protezione: *Generalization* e *Suppression*, le quali preservano la veridicità dei dati (diversamente da swapping e scrambling).

1.1.1 *Generalization*

Sostituzione dei valori di un attributo con valori più generali. Consideriamo:

- *Domain*: set di valori che un attributo può assumere.
- *Generalized domains*: contiene valori generalizzati e relativo mapping tra ogni domain e ogni sua generalizzazione.
- `Dom`: set di domini originali con le loro generalizzazioni.
- *Generalization relationship* $\leq_D$: dati D_i, $D_j \in$ `Dom`, $D_i \leq D_j$ significa che i valori in D_j sono generalizzazioni dei valori in D_i.

  $\leq_D$ definisce ordinamento parziale su `Dom` ed è richiesto nelle seguenti condizioni:

Condizione 1 (C1 - Determinismo nel processo di generalizzazione)
$\forall D_i, D_j, D_z$ *Dom:*
$D_i \leq_D D_j$, $D_i \leq_D D_z \implies D_j \leq_D D_z \vee D_z \leq_D D_j$[2].

Condizione 2 (C2 -)
Tutti gli elementi massimali di `Dom` *sono singoletti (singleton)* [3].

- **DGH**$_D$ - *Domain Generalization Hierarchy*: gerarchia di ordninamento totale per ogni dominio $D \in$ `Dom`.

Per quanto riguarda i valori nei domini consideriamo:

- *Value generalization relationship* $\leq_V$: associa ogni valore in D_i ad un unico valore in D_j, sua generalizzazione.
- **VGH**$_D$ - *Value Generalization Hierarchy*: albero dove
  - Foglie sono valori in D.
  - Radice è il valore, singolo, nell'elemento massimale di DGH$_D$

[2] Questo comporta che ogni dominio D_i ha al massimo un solo dominio di generalizzazione diretta D_j

[3] La condizione assicura che tutti i valori in ogni dominio possano essere generalizzati ad un singolo valore

1.1.2 *Suppression*

Consideriamo Soppressione di Tupla. Questa "modera" la *Generalization* quando un numero limitato di *outlier*[4] forzerebbe una generalizzazione elevata.

1.2 Generalizzazione *k*-Minima

Definizione 3 (Table Generalizzata con Soppressione)
Consideriamo T_i e T_j due table sugli stessi attributi.
T_j è generalizzazione (con soppressione di tupla) di T_i, riportata come $T_i \preceq T_j$, se:

1. $|T_j| \leq |T_i|$
2. *Dominio $dom(A, T_j)$ è uguale o una generalizzazione di $dom(A, T_i)$, dove A indica ogni attributo in $T_{i,j}$*
3. *E' possibile definire funzione iniettiva che associa ogni tupla $t_j \in T_j$ con una tupla $t_i \in T_i$, per la quale ogni attributo in t_j è uguale o generalizzazione del corrispondente in t_i.*

Definizione 4 (Distance Vector)
Siano $T_i(A_1, ..., A_n)$ e $T_j(A_1, ..., A_n)$ tali che $T_i \preceq T_j$.
il distance vector di T_j da T_i è il vettore
$DV_{i,j} = [d_1, ..., d_n]$
dove ogni $d_z, z = 1, ..., n$ è la lunghezza dell'unico percorso tra $dom(A_z, T_i)$ e $dom(A_z, T_j)$ nella DGH_{D_z}

Corollario 1 (Ordine Parziale tra DV)
$DV = [d_1, ..., d_n] \leq DV' = [d'_1, ..., d'_n]$ se e solo se $d_i \leq d'_i$ per $i = 1, ..., n$.

Si costruisce una *gerarchia di distance vectors* come lattice (diagramma) corrispondente alla DGH_D come in fig. 1.1

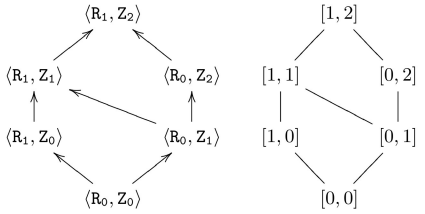


Figure 1.1

[4]Secondo paper riportato, per *Outlier* si intende respondent che sono "molto diversi" o "molto distanti" dalla maggior parte degli altri individui. DOI: `https://doi.org/10.1007/978-3-662-46497-7_11` .

Per bilanciare tra perdita di precisione dovuta a *Generalization* e perdita di completezza dovuta a *Suppression* si suppone che data holder determini la soglia `MaxSup`, che indica il numero di tuple che possono essere soppresse.

Definizione 5 (Generalizzazione *k*-minima con Soppressione)
Siano T_i e T_j due table tali che $T_i \preceq T_j$, e sia `MaxSup` *la soglia di soppressione accettabile scelt. T_j è una generalizzazione k-minima di T_i se e solo se:*

1. *T_j soddisfa k-anonimity applicando soppressione minima, ossia T_j soddisfa k-anonimity e: $\forall T_z : T_i \preceq T_z$, $DV_{i,z} = DV_{i,j}$, T_z soddisfa k-anonimity $\implies |T_j| \geq |T_z|$.*
2. *$|T_i| - |T_j| \leq$ `MaxSup`.*
3. *$\forall T_z : T_i \preceq T_z$ e T_z soddisfa le condizioni 1 e 2 $\implies \neg(DV_{i,z} < DV_{i,j})$.*

Ultima espressione rende meglio come $DV_{i,z} \geq DV_{i,j}$. Il concetto che esprime è che *"non esiste un'altra Generalization T_z che soddisfi 1 e 2 con un DV minore di quello di T_j"*

Diversi **preference criteria** possono essere applicati nella scelta della generalizzazione minimale preferita:

- **Distanza assoluta minima**: minor numero totale di passi di generalizzazione (indipendentemente dalle gerarchie di *Generalization* considerate).
- **Distanza relativa minima**: minimizza il numero relativo di passi di generalizzazione (passo relativo ottenuto dividendo per l'altezza del dominio della gerarchia a cui si riferisce.
- **Massima distribuzione**: maggior numero di tuple distinte.
- **Minima soppressione**: minor tuple soppresse (maggior cardinalità).

1.3 Classificazione tecniche di k-anonimity

Classificazione delle tipologie di composizione *Generalization* e *Suppression* sono mostrati in fig.1.2.

	Suppression			
Generalization	*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

Fig. 8. Classification of k-anonymity techniques

Figure 1.2

Casi *not applicable* (**CG_TS** e **CG_AS**): supportare *Generalization* a grana fine (cella) implica poter applicare soppressione allo stesso livello.

Vengono esposte qui di seguito alcune soluzioni algoritmiche in letteratura, riassunte in fig.1.3

Algorithm	**Model**	**Algorithm's type**	**Time complexity**
Samarati [26]	AG_TS	Exact	exponential in $\|QI\|$
Sweeney [29]	AG_TS	Exact	exponential in $\|QI\|$
Bayardo-Agrawal [5]	AG_TS	Exact	exponential in $\|QI\|$
LeFevre-et-al. [20]	AG_TS	Exact	exponential in $\|QI\|$
Aggarwal-et-al. [2]	_CS	$O(k)$-Approximation	$O(kn^2)$
Meyerson-Williams [24][2]	_CS	$O(k \log k)$-Approximation	$O(n^{2k})$
Aggarwal-et-al. [3]	CG_	$O(k)$-Approximation	$O(kn^2)$
Iyengar [18]	AG_TS	Heuristic	limited number of iterations
Winkler [33]	AG_TS	Heuristic	limited number of iterations
Fung-Wang-Yu [12]	AG_	Heuristic	limited number of iterations

Figure 1.3: Alcuni approcci a k-anonimity(n è numero di tuple in PT).

1.4 Algoritmo Samarati (AG_TS)

Il primo algoritmo per garantire k-anonimity è stato proposto insieme alla definizione di k-anonimity. La definizione di k-anonimity è basata sul QI quindi l'algoritmo lavora solo su questo set di attributi e su table con più di k tuple.

Data una *DGH* ci sono diversi percorsi dall'elemento in fondo alla gerarchia alla radice. Ogni percorso è una differente *strategia* di generalizzazione. Su ogni percorso c'è esattamente una *Generalization* minima localmente (nodo più basso che garantisce k-anonimity).

In maniera naif si può cercare su ogni percorso il minimo locale per poi trovare il minimo globale tra questi ma non è praticabile per l'elevato numero di percorsi.

Per ottimizzare la ricerca si sfrutta la proprietà che salendo nella gerarchia la soppressione richiesta per avere k-anonimity diminuisce:

- Ogni nodo in DGH viene associato ad un numero, **height**, corrispondente alla somma degli elementi nel Distance Vector associato.
- Altezza di ogni DV nel diagramma (*distance vector lattice* `VL`) si scrive come $height(DV,\texttt{VL})$.

Se non c'è soluzione che soddisfi k-anonimity sopprimendo meno di `MaxSup` ad altezza h non può esistere soluzione che soddisfi ad una altezza minore.

L'algoritmo usa binary search cercando la minore altezza in cui esiste un DV che soddisfa k-anonimity rispettando `MaxSup` e ha come primo passo:

$$\text{Cerco ad altezza } \lfloor \frac{h}{2} \rfloor : \begin{cases} \text{trovo vettore che soddisfa } k\text{-anonimity} \implies \lfloor \frac{h}{4} \rfloor \\ \text{altrimenti} \implies \text{cerco in } \lfloor \frac{3h}{4} \rfloor \end{cases} \tag{1.1}$$

La ricerca prosegue fino a trovare l'altezza minore in cui esiste vettore che soddisfa k-anonimity con `MaxSup`.

1.4.1 Evitare il calcolo delle table generalizzate

Algoritmo richiederebbe il calcolo di tutte le table generalizzate. Per evitarlo introduciamo il concetto di DV tra tuple.

Definizione 6 (Distance Vector tra tuple - Antenato Comune)
Sia T una table.
Siano $x, y \in T$ due tuple tali che $x = \langle v'_1, ..., v'_n \rangle$ e $y = \langle v''_1, ..., v''_n \rangle$ con v'_i e v''_i valori in D_i con $i = 1, ..., n$.
*Il **distance vector** tra x e y è $V_{x,y} = [d_1, ..., d_n]$. dove d_i è la lunghezza (uguale) dei due percorsi da v'_i e v''_i al loro comune antenato comune più prossimo v_i sulla VGH_{D_i}.*

In altri termini ogni distanza in $V_{x,y}$ è una distanza uguale dal dominio di v'_i e v''_i al dominio in cui sono generalizzati allo stesso valore v_i.

Allo stesso modo $V_{x,y}$ per $x, y \in T_i$ equivale a $DV_{i,j}$ per $T_i \preceq T_j$ per cui x e y vengono generalizzate alla stessa tupla t.

Per il momento il resto è delirio TODO

1.5 Bayardo-Agrawal: *k-Optimize* (AG_TS)

Approccio considera che la generalizzazione di attributo A su dominio **ordinato** D corrisponde ad un partizionamento del dominio dell'attributo in intervalli. Ogni valore del dominio deve comparire in un intervallo e ogni valore in un intervallo precede ogni valore degli intervalli che lo seguono.

Si assume quindi un ordine tra gli attributi del *Quasi-Identifier*. Inoltre associa un valore intero chiamato *index* ad ogni intervallo di ogni dominio degli attributi del *Quasi-Identifier*.

Una *Generalization* è quindi rappresentata come l'unione dei valori di indice per ogni attributo (il valore più piccolo in un dominio viene omesso perchè comparirà sicuramente nella generalizzazione per quel dominio).

k-Optimize costruisce un *set enumeration tree* sul set I di valori di indice, con radice vuota. Ogni nodo figlio è costruito dal padre inserendo in coda un index di I maggiore degli altri index già presenti nel padre (per ordinamento totale).

La visita dell'albero permette di valutare con Depth First Search ogni nodo, prunando ogni nodo e tutti i suoi figli se questi non possono corrispondere a soluzioni ottimali. Nello specifico *k-Optimize* pruna un nodo n quando determina che nessuno dei suoi discendenti può essere ottimale. Data una funzione di costo l'algoritmo calcola un limite inferiore sul costo che può essere ottenuto sul subtree del nodo n, prunando se nodo a costo minore è già stato trovato.

Se un nodo viene prunato allora anche altri nodi, non per forza del sottoalbero, possono essere prunati: supponendo di prunare il nodo {1,3} in figura 1.4 allora posso prunare qualunque altro nodo contenente 1 E 3, come ad esempio {1,2,3}.

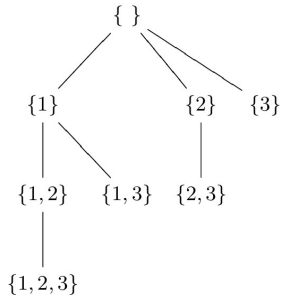


Figure 1.4: Set enumeration tree sull'insieme di indici I={1,2,3}

1.6 LeFevre-DeWitt et al.: Incognito (AG_TS)

Idea di base è riassunta nella seguente definizione:

Definizione 7 (*k*-anonimity dei subset di QI)
Se una table T con Quasi-Identifier QI composto da $m = |QI|$ attributi soddisfa k-anonimity, T soddisfa k-anonimity anch per qualunque Quasi-Identifier QI' talce che $QI' \subset QI$.

Pertanto k-anonimity su un subset di QI è condizione necessaria (e non sufficiente) per k-anonimity di T sull'intero QI.

Algoritmo esclude in anticipo alcune generalizzazioni della gerarchia con un calcolo a priori.

Strategia: bottom-up BFS sulla DGH. Incognito genera tutte le possibili table k-anonime minime secondo i seguenti passaggi:

1. (Iterazione 1): verifica k-anonimity per ogni singolo attributo nel *Quasi-Identifier*, scartando quelle che non soddisfano.
2. (Iterazione 2): combina a coppie le *Generalization* non scartate al passo 1 verificando k-anonimity.
3. (...)
4. (Iterazione m): arriva a considerare l'intero set di attributi di QI

Utilizzando approccio bottom-up, per la condizione citata prima, se una *Generalization* soddisfa k-anonimity allora anche sue ulteriori dirette generalizzazioni soddisfano k-anonimity e pertanto non sono considerate ulteriormente.

Un esempio per $QI = \{Race, Sex, Marital\,Status\}$ è riportato nella figura 1.5 che segue:

1.7 Algoritmi Euristici

k-anonimity è un problema NP-difficile e pertanto ha complessità esponenziale nella dimensione del QI.

Alcuni approcci euristici alternativi proposti:

- Iyengar - algoritmi genetici e ricerca stocastica incompleta.
- Winkler - "simulated annealing" che non garantisce qualità ed è costoso.
- Fung, Wang, Yu - Top-down che, partendo dalla soluzione più generale e specializza iterativamente alcuni valori della soluzione corrente finchè non viola k-anonimity[5].

Essendo metodi euristici non esiste un limite di efficienza per queste soluzioni.

[5] Algoritmo determina ad ogni passo una "buona" specializzazione guidata da una metrica che compara sia *information gain* che *anonimity loss*

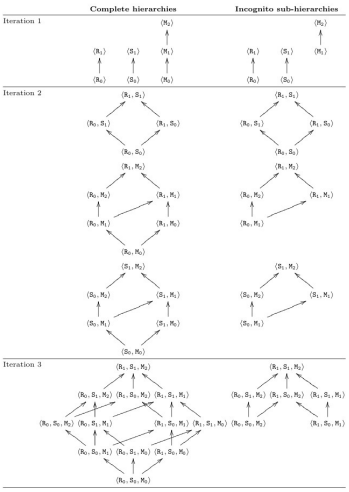


Figure 1.5

1.8 Algoritmi per _CS e CG_

Algoritmi esatti per _CS e CG_, operando a livello di cella, possono avere costo computazionale esponenziale nel numero di tuple della table. Per questo motivo si sfruttano algoritmi di approssimazione.

La descrizione degli algoritmi è qui omessa in quanto non presentati a lezione, ma indicazione sul nome e sul costo computazionale sono riportati in fig.1.3.

1.9 Ulteriori studi riguardo *k*-anonimity

Breve resoconto di alcuni studi riguardanti k-anonimity.

1.9.1 *k*-anonimity multidimensionale

Nella proposta originale di k-anonimity si assume che ogni valore abbia una sola generalizzazione. Tuttavia un *Generalization* step potrebbe portare a diversi valori generalizzati (e.g. 91412 $\rightarrow$ 9141* *oppure* 914*2) e quindi avremmo, invece che un albero, un grafo per la DGH.

LeFevre, DeWitt e Ramakrishnan hanno proposto un modello multidimensionale NP-difficile con approssimazione greedy, sia per domini numerici che categorici. Complessità è $O(n \log n)$ con n numero di tuple, e la qualità della GT è migliore rispetto alla monodimensionale.

1.9.2 *l*-diversity

Qui riportata l-diversity come presentata a lezione, in considerazione dell'*homogeneity attack* e del *background knowledge attack*. Autori del concetto di l-diversity sono Machanavajjhala, Gehrke, and Kifer.

1.9.3 Valutazione della *k*-anonimity

Analizzando il risultato della k-anonimity con mining, Aggarwal mostra che all'aumentare della dimensione $|QI|$ la perdita di informazione potrebbe diventare elevata (perchè la probabiità che k tuple siano simili è molto bassa).

1.9.4 Algoritmi distribuiti

Anonimizzare dati distribuiti tra diverse parti interconnesse. Diverse proposte:

- Jiang e Clifton: partizione verticale della table e storage in siti diversi. Ricostruzione con join tramite una chiave comune. Necessaria strategia di k-anonimity comune.
- Wang, Fung e Dong: come Jiang ma strategia negoziata interattivamente tra le parti.
- Zhong, Yang e Wright: partizione orizzontale e due soluzioni di cui una prevede crypto valori sensibili e l'altra prevede _CS.

1.9.5 *k*-anonimity con viste multiple

Definizione 8 (Data Association)
Due o più attributi sono da considerare più sensibili quando i loro valori sono associati rispetto a quando questi valori appaiono separatamente.

Un problema di questo tipo risulta evidente nell'esempio in fig.1.6 in cui è facile ricostruire le associazioni `John_Obesity` e `David_Obesity` dalle viste v_1 e v_2.

Name	Sex	Disease
Sue	F	hypertension
Claire	F	obesity
Ann	F	chest pain
John	M	obesity
David	M	obesity
Mary	F	short breath
Alice	F	short breath
Carol	F	chest pain
Kate	F	short breath

PT

Name	Sex
Sue	F
Claire	F
Ann	F
John	M
David	M
Mary	F
Alice	F
Carol	F
Kate	F

view v_1

Sex	Disease
F	hypertension
F	obesity
F	chest pain
F	short breath
M	obesity

view v_2

Figure 1.6: Una `PT` e due sue possibili viste v_1 e v_2

Yao, Wang e Jajodia mostrano che il problema è NP^{NP}-difficile, mentre se non ci sono dipendenze funzionali tra le viste esiste soluzione polinomiale.

1.9.6 *k*-anonimity con Micro-Aggregazione

Domingo-Ferrer e Mateo-Sanz propongono di usare Micro-Aggregation al posto di *Generalizatione* *Suppression*.

Definizione 9 (Micro-Aggregation)
Micro-Aggregation consiste nel:

1. *dividere microdata in cluster di almeno k tuple ciascuno.*
2. *per ogni attributo, il valore medio sul cluster sostituisce il valore della tupla nel cluster.*

La partizione ottimale è quella che massimizza l'homogeneity nel cluster, ossia clusterizza valori simili. Questo riduce l'information loss.

La somma dei quadrati è il metodo tradizionale per determinare il cluster.

Essendo NP-difficile, si riduce complessità riducendo lo spazio delle soluzioni a cluster di dimensione tra k e $2k$[6].

1.9.7 *k*-anonimity per la Location Privacy

Solzioni che garantiscono k-anonimity non tra tuple in un database ma in un set di individui che inviano un messaggio nello stesso contesto spazio-temporale.

1.9.8 *k*-anonimity per i Protocolli di Comunicazione

Un protocollo di comunicazione è *sender k-anonymous* (*receiver k-anonymous*) se l'attaccante può rilevare solo un set di k possibili *sender (receiver)* er un messaggio.

[6] k per garantire k-anonimity, $2k$ per ridurre information loss.

Chapter 2

Protezione Microdati

2.1 Introduzione

2.2 Macrodata vs Microdata

2.3 Classificazione delle tecniche di protezione riguardo il Disclosure dei microdati

Il controllo del disclosure dei microdati è un'importante problema pratico sia nel privato che nel pubblico.
La protezione dei microdati ha apparentemente due obiettivi tra di loro contrastanti. Da una parte si vuole evitare la re-identificazione, la quale accade ogni qualvolta l'informazione del respondent che appare nella table di microdati è identificata; il quale, è associata con le identità dei respondent corrispondenti.

Dall'altra invece le applicazioni di tecniche dovrebbero preservare le *chiavi delle proprietà statistiche* dei dati originali i quali sono state segnati come importanti dai destinatari dei dati (ossia chi riceverà i dati finali)

Precisamente data una table di microdati T, una tecnica di protezione dei dati dovrebbe trasformare questa table T in una table T_1 in modo che:

1. il rischio che un utente malevolo possa usare T_1 per determinare informazioni confidenziali o identificare un respondent, sia basso

2. l'analisi statistica su T e T_1 possa produrre risultati simili

Generalmente i seguenti fattori contribuiscono al rischio di disclosure:

- L'esistenza di tuple altamente visibili (ossia tutte quelle tuple con caratteristiche che hanno caratteristiche uniche)

- Possibilità di matching tra le table di microdati e le informazioni esterne.
- L'esistenza di un alto numero degli attributi comuni che possono aumentare la possibilità di linking.

Mentre i fattori che minimizzano il rischio di disclosure sono:

- Una table di microdati contenenti un subset della popolazione intera. Questo implica che le informazioni di un respondent specifico, il quale potrebbe essere un utente malevolo potrebbe voler sapere, potrebbe non essere incluso nella table di microdati.
- Le informazioni specificate nelle table rilasciate, quindi pubbiche non sono aggiormate. Potrebbero cambiare nel tempo, indi per cui il linking con le informazioni esterne potrebbero non essere accurati
- Una table di microdati e le informazioni esterne contengono rumore, riducendo la probabilità di linking delle informazioni
- Una table di microdati e le risorse esterne possono contenere dati espressi in forme diverse, riducendo l'abilità di linking

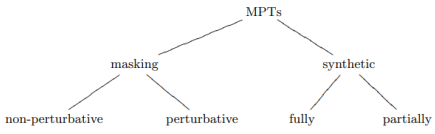


Fig. 3. Classification of microdata protection techniques (MPTs)

Figure 2.1: Classificazione delle tecniche di protezione dei microdati

Generalmente, per limitare il rischio di disclosure di una table di microdati è necessario sopprimere gli identifiers impliciti ed espliciti come step iniziale (e.g. SSN, Name)

Questo processo è noto come de-identificazione, quest'ultima non necessariamente rende le tuple anonime, siccome è possibile fare re-identificazione usando informazioni esterne.

Tipicamente le tecniche sono basate sul principio che la re-identificazione possa essere contrastata riducendo la quantità delle informazioni rilasciate, mascherando i dati (e.g., non rilasciando o perturbando i valori), o rilasciando valori plausibili modificati invece di quelli reali. Seguendo questo principio le tecniche si suddividono in duo macro categorie:

- Tecniche di mascheramento: I dati originali sono trasformati producendo nuovi dati che non sono validi per le analisi statistiche in modo da preservare la confidenzialità dei respondent. Le tecniche di mascheramento si suddividono in:

  - non perturbative: il dato originale non viene modificato, ma alcuni dati vengono soppressi o vengono rimossi dettagli.
  - perturbativi: i dati originali sono modificati.

- Generazione dei dati sintetici: I set di tuple originale nelle table di microdati sono sostituiti da un nuovo set di tuple generati in modo di preservare le proprietà statistiche chiave del dato originale.Dal momento che le table dei microdati rilasciati contengono dati sintetici, il rischio di reidentificazione viene ridotto. Questi possono essere interamente sintetici (fully synthetic) o misti con i dati originali (partially synthetic).

  Un'altra caratteristica importante delle tecniche di protezione è che possono essere usati su tipo di dati differenti. In particolare i tipi di dati possono essere categorizzati come

  - *Continui*: Un attributi è detto continuo se le operazioni numeriche e aritmetiche sono definite sull'attributo. E.g. Data di nascita e temperatura sono attributi continui.
  - *Categorici*: Un attributo è definito categorico se le operazioni aritmetiche non hanno alcun senso se fatto sull'attributo. E.g. Razza, su cui non si possono fare operazioni aritmetiche

Di seguito, sono descritti le tecniche di protezione dei microdati principali, e se sono applicabili a dati continui, categorici o entrambi.

Technique	**Continuous**	**Categorical**
Sampling	yes	yes
Local suppression	yes	yes
Global recoding	yes	yes
Top-coding	yes	yes
Bottom-coding	yes	yes
Generalization	yes	yes

Figure 2.2: Applicazione delle tecniche non perturbative su dati differenti

Technique	**Continuous**	**Categorical**
Resampling	yes	no
Lossy compression	yes	no
Rounding	yes	no
PRAM	no	yes
MASSC	no	yes
Random noise	yes	yes
Swapping	yes	yes
Rank swapping	yes	yes
Micro-aggregation	yes	yes

Figure 2.3: Applicazione delle tecniche perturbative su dati differenti

2.4 Tecniche di Mascheramento

Alcune tecniche di mascheramento

2.4.1 Tecniche non perturbative

Le tecniche non perturbative producono microdati eliminando dettagli dai dati originali. Di seguito alcune di esse:

- *Sampling*: I table dei microdati protetti includono solo tuple formate da sample (campioni) della popolazione totale. Siccome c'è dell'incertezza se uno specifico respondent sia presente nei sample, il rischio di re-identificazione viene riudotto. Per esempio possiamo decidere se pubblicare solo le tuple pari della table originale. Questa tecnica può operare solo su dati categorici

- *Local suppression*: Sopprime i valori degli attributi in modo da limitare la possibilità di analisi. Questa tecnica lascia uno spazio vuoto su alcuni attributi (sensitive cells) che contribuiscono in modo significante al rischio di disclosure delle tuple coinvolte.

- *Global Recoding*: Il dominio degli attributi, ossia tutti i possibili valori, sono divisi intervalli disgiunti di grandezza uguale, e ogni intervallo viene associato a una label. La table dei microdati protetti sono ottenuti sostituendo i valori degli attributi con le label associate agli intervalli corrispondenti. Questa tecnica riduce i dettagli della table e quindi riduce il rischio di re-identificazione. Sono presenti due tecniche particolari di recoding, ossia *Top-Coding* e *Bottom-Coding*, qui sotto descritte:

  - *Top-Coding*: E' basata sulla definizione di limite superiore (upper limit), chiamato top-code, per ogni attributo che deve essere protetto. Ogni valore maggiore di questo valore è sostituito dal top-code. Questa tecniche può essere applicata agli attributi categorici che possono essere ordinati linearmente come gli attributi continui.
  - *Bottom-Coding*: Simile al top-coding ma definisce il limite inferiore (lower limit). Quindi, ogni valore più basso di questo non verrà ripubblicato e verrà sostituito con il bottom-code.

- *Generalization*: Consiste nel rappresentare i vaori di un dato attributo usando valori più generali. Questa tecnica è basata sulla definizione di generalizzazione gerarchica, dove i valori più generali sono alla radice della gerarchia e le foglie corrispondono ai valori specifici. Un processo di generalizzazione perciò procede con la sostituzione dei valori rappresentati dai nodi foglia con dei valori al nodo superiore.

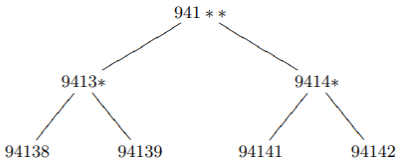


Fig. 6. Generalization hierarchy for attribute ZIP

Figure 2.4: Generalizzazione gerarchica

2.4.2 Tecniche perturbative

Con le tecniche perturbative le table dei microdati vengono modificate per la pubblicazione Le modifiche possono portare a combinazioni dei valori originali uniche le quali svaniscono non appena vengono introdotte nuove combinazioni.

- *Resampling*: Questa tecnica sostituisce i valori degli attributi sensibili continui con un valore medio computato sul numero di samples della popolazione originaria. Precisamente, assumiamo che N sia il numero di tuple presenti in una table di microdati, e $S_1, ..., S_t$ con t i sample di dimensione N. Ogni sample viene rankato indipendentemente e successivamente viene calcolata la media del j_{esimo} valore ranked. I valori medi ottenuti sono riordinati prendendo in considerazione l'ordine dei valori originali, seguendo quindi l'ordinamento della tabella iniziale.

Microdata Protection

S_1	S_2	S_3	S_4
260	220	170	210
170	280	290	190
200	210	220	230
280	310	270	200
190	290	185	185
185	180	300	260
200	285	250	220
290	265	260	290
170	150	190	230
300	270	270	310
200	298	200	170

(a) Initial samples

S_1	S_2	S_3	S_4	Average
170	150	170	170	165
170	180	185	185	180
185	190	190	190	188.75
190	210	200	200	200
200	220	220	210	212.5
200	265	250	220	233.75
200	270	260	230	240
260	280	270	230	260
280	285	270	260	273.75
290	290	290	290	290
300	310	300	310	305

(b) Ordered samples

Original value (S_1)	Released value
260	260
170	165
200	212.5
280	273.75
190	200
185	188.75
200	233.75
290	290
170	180
300	305
200	240

(c) Released data

Fig. 8. An example of resampling over attribute Chol

Figure 2.5: Esempio di resampling

- *Rounding*: Simile alla tecnica usata anche per la protezione dei macrodati ed è applicabile solo agli attributi continui. Sostituisce i valori originali con dei valori approssimati, quest'ultimi sono scegli da un set di *rounding points* p_i ognuno il quale definisce il *rounding set*. Per esempio:
  - ***Rounding Points*** possono essere scelti come multipli di un valore base b, con $b = p_{i+1} - p_i$.
  - ***Rounding Sets*** definiti come
    * $\begin{cases} [p_i - \frac{b}{2}, p_i + \frac{b}{2}) \text{con } i = 2, ..., r-1 \\ [0, p_1 + \frac{b}{2}) \\ [p_r - \frac{b}{2}, X_{max}], \text{con } X_{max} \text{ il valore più grande possibile} \end{cases}$
    * rispettivamente per p_1 e p_r
    * Un valore v di X viene sostituito dal round point corrispondente al rounding set dove è contenuto v.

- *Random Noise*: Perturba gli attributi sensibili aggiungendo o moltiplicando una variabile casuale con una distribuzione. Il rumore additivo (additive noise) è preferito al rumore moltiplicativo (multiplicative noise) e viene formalmente definito così:
  Poniamo X_j come la j_{esima} colonna della table dei microdati originaria corrispondente a un attributo sensibile e supponiamo che ci siano N tuple. Ogni valore X_{ij}, con $i = 1, ..., N$, viene sostituito con $x_{ij} + \varepsilon_{ij}$, dove ε_j è il vettore degli errori normalmente distribuiti ricavati da una variabile casuale con la media uguale a 0, in generale, con una varianza che sia proporzionale agli attributi originari

  Forse da approfondire
  [1]

- *Swapping*: Consiste nel modificare il subset di una table di microdati facendo lo swapping dei valori di un set formato degli attributi sensibili, chiamato *swapped attributes*, tra coppie di tuple selezionate (le coppie sono selezionate secondo un criterio ben definito). Intuitivamente, questa tecnica riduce il rischio di re-identificazione perché introduce dell'incertezza sul valore reale legato al dato del respondent. Anche se questa tecnica è facile da applicare, generalmente ha lo svantaggio di non preservare le proprietà dei sottodomini. Inizialmente la tecnica originale fu presentata solamente per gli attributi categorici.

[1] N.B ϵ viene usato per perturbare i dati ma lasciando intatta la correlazione con la tabella iniziale

- *Micro-Aggregation (o Blurring)*: Consiste nel raggupprare le tuple individiduali in piccole aggregazioni di una dimensione fissata k: verrà pubblicata la media di ogni aggregato invece dei valori individiduali. Anche se esistono diverse funzioni per caloclare la similrità, può essere difficile trovare un soluzione di raggruppamento ottimale. Ci sono diverse varianti di *micro-aggregation*, per esempio la media può sostituire il valore originale anche solo per una singola tupla nel gruppo o per tutte. Molti attributi possono essere protetti grazie a *micro-aggregation* usando lo stesso o un diverso raggruppamento e, la dimensione del gruppo puo essere variabile o fissata ad un minimo.

2.5 Tecniche per la Generazione di Dati Sintetici

La generazione dei dati sintetici è una valida alternaitiva per la protezione dei microdati. Il principio base su cui queste tecniche sono basate riguarda la non correlazione tra i contenuti statistici dei dati e l'informazione provvista da ogni respondent, quindi un modello ben rappresentante il dato potrebbe sostiture il dato stesso. Un requisito molto importante per la generazione dei dati sintetici, che rende il processo di generazione un problema importante, è che i dati sintetici e il dato originale potrebbero presentare la stessa qualità delle analisi statistiche. Il vantaggio maggiore di questi tipi di tecniche è che i dati sintetici rilasciati non sono collegati a nessun respondent, indi per cui il rilascio non può essere soggetto al fenomeno di re-identificazione; questo, inoltre, lascia ai proprietari dei dati di porre maggiore attenzione sulla qualità del dato rilasciato, invece che al problema di re-identificazione.

2.6 Misure per valutare la confidenzialità e l'utilità dei Microdati

Come discusso precedentemente, ci sono svariate tecniche per la protezione dei microdati. Le performance di queste tecniche di protezione sono valutate in termini di *information loss* e *disclosure risk*.

- Information loss è la quantità di informazioni che esistono nei microdati originali e che, a causa delle tecniche di protezione, questo non si verifica nei microdati protetti.

- Disclosure risk è il rischio di incorrere nel disclosure quando i micordati vengono rilasciati.

Esistono due soluzioni estreme per il rilascio di microdati, le quali sono:

- Crittare i dati originali (rimuove il rischio di disclosure e il massimo di information loss)

- Rilasciare i dati originali (massimo rischio di disclosure e nessun information loss)

In questa sezione andremo a descrivere alcuni dei pmetodi più importanti usati per quantificare il rischio di disclosure e di perdita di informazione (information loss).

2.6.1 Rischio di Disclosure (Disclosure Risk)

Generalmente ci sono due tipi di disclosure: *disclosure di identità* e*disclosure di attributi*. Il disclosure riguardo l'identita si occupa di una specifica identità la quale può essere linkata a tuple presenti nelle table di microdati. La disclosure riguardo gli attributi invece l'informazione che può essere spifferata di un attributo legato ad un individuo. In generale, due fattori che potrebbero avere un impatto sul disclosure di identità sono:

- *univocità della popolazione*, ossia la probabilità di identificare un respondent, il quale è unico, con una specifica combinazione di attributi è elevata se quei attributi sono presenti nella table di microdati.
- *re-identificazione*, ossia, che i mcirodati rilasciati sono linkabili ad un'altra table pubblicata, dove gli *identifiers* non sono stati rimossi.

Sono stati proposti diversi metodi per misurare il rischio di disclosure dei microdati rilasciati. Per esempio, *la combinazione minima non sicura degli attributi* ritorna il numero degli attibuti con una combinazione unica presenti in una specifica tupla di microdati. Questo metodo può essere adottato solo con le tecniche di mascheramento non perturbative, e, più è alto questo valore tanto più basso sarà il rischio di disclosure.

Univocità
Ogni qual volta un sample unico è anche una popolazione unica, il disclosure d'identità diventa molto più probabile. Esistono metodi differenti per valutare il rischio di univocità, e tutti i metodi si affidano alla valutazione della probabilità. Il primo metodo misura la probabilità dell'univocità della popolazione (*population uniqueness (PU)*), ossia la probabilità che nella popolazione esista solo un individuo con una certa combinazione di valori su un determinato insieme di attributi. Questa probabilità è misurata come:
$Pr(PU) = \sum_j I$, con $\frac{F_j=1}{N}$, dove N è la dimensione della popolazione e F_j è il numero degli individui nella popolazione con la j_{esima} combinazione sugli attributi considerati, e $I()$ è una funzione e:
$\begin{cases} I(A) = 1, \text{ se } A = true \\ 0, \text{ altrimenti.} \end{cases}$

Il secondo metodo misura la probabilità che un sample unico *sample unique (SU)* sia anche una popolazione unica *(PU)*. questa probabilità è misurata come:

$PR(PU|SU) = \frac{\sum_j I(f_j=1, F_j=1)}{\sum_j I(f_j=1)}$, dove f_j è il numero degli individui presenti nei sample con la j_{esima} combinazione sugli attributi considerati.

Questi metodi sono chiamati misure *file-level* perché assegnano lo stesso rischio a tutte le tuple. Il rischio di disclosure *tuple-level* è la probabilità di disclosure dell'identità di un individuo specifico.
Questi tipi di misure sono state introdotte a causa della non-omogeneità del rischio di re-identificazione su un'intera tabella di microdati.
Supponendo che ci siano K combinazioni differenti dei valori legati ai quasi-identifiers. Queste combinazioni producono una suddivisione sia nella popolazione che nei sample. Assumendo che F_k sia la frequenza della k_{esima} suddivisione, il rischio di disclosure per una tupla presente in un sample con la k_{esima} combinazione è $\frac{1}{F_k}$.
Il problema di questo metodo è che F_k non è generalmente noto per la popolazione. Poiché le frequenze della distribuzione campionaria f_k sono note, si considera la distribuzione delle frequenze F_k, data f_k ($F_k|f_k$ può essere modellata come una binomiale negativa).

Si noti che l'univocità può essere utilizzata come misura del rischio di divulgazione solo solo se i microdati sono stati protetti attraverso una tecnica di mascheramento non perturbativa.
Le tecniche perturbative modificano i valori dei dati e quindi non è possibile stabilire correttamente la frequenza di un valore nel campione rilasciato, perché possono essere introdotte nuove combinazioni uniche e possono scomparire le combinazioni uniche originali.

Record Linkage
Consiste nel trovare un match tra le tuple presenti nelle table di microdati protetti e le tuple presenti nelle fonti esterne di informazione pubbliche e non anonime. Poiché non è possibile conoscere a priori tutte le fonti esterne di informazione che possono essere utilizzate da un eventuale utente malintenzionato, viene eseguito un controllo probabilistico sui microdati protetti.
È necessario adottare diversi metodi di record linkage a seconda che la tabella di microdati e le informazioni esterne abbiano o meno attributi comuni. Se ci sono attributi comuni, è necessario innanzitutto adottare una rappresentazione unica per gli attributi comuni. Ad esempio, abbreviazioni diverse nel nome di una persona porterebbero a concludere che due tuple non sono correlate, mentre in realtà si riferiscono allo stesso respondent. È quindi possibile adottare una strategia per il record linkage. I metodi di record linkage possono essere suddivisi in tre grandi categorie:

- *deterministici*: Cerca una corrispondenza esatta su uno o più attributi tra tuple di diversi set di dati. Il principale svantaggio di questo metodo è che non prende in considerazione la rilevanza dell'attributo nel trovare un collegamento.

- *probabilistici*: Dati due dataset, D_1 e D_2, viene calcolato l'insieme di tutte le possibili coppie di tuple (d_{1i}, d_{2j}), dove
  $\begin{cases} d_{1i} \in D_1 \\ d_{2j} \in D_2. \end{cases}$

  A ogni coppia è associata una probabilità che rappresenta se la coppia è una corrispondenza reale.

  - Se la probabilità è inferiore a una soglia fissa T_1, la coppia viene scartata perché le tuple sono considerate non linked
  - Se la probabilità è superiore a una seconda soglia fissa T_2, la coppia è considerata una corrispondenza reale
  - Se la probabilità è compresa tra T_1 e T_2, è necessaria una valutazione umana per verificare se rappresenta una corrispondenza o meno. Tale probabilità viene calcolata considerando diversi pesi per i vari attributi e l'accordo o l'accordo parziale sui valori degli attributi. I pesi associati agli attributi e le due soglie T_1 e T_2 sono stabiliti dal proprietario dei dati.

- *basati sulla distanza*: Dati due insiemi di dati, D_1 e D_2, ogni tupla $d_{1i} \in D_1$ viene abbinata alla tupla più vicina $d_{2j} \in D_2$.
  Questo metodo richiede la definizione di una funzione di distanza f tra coppie di tuple. Per esempio, la definizione di f può sfruttare funzioni di distanza definite sugli attributi e può assegnare pesi diversi a ciascun attributo, a seconda della sua importanza nel processo di link. Un esempio di funzione di distanza è la *Distanza Euclidea*, che considera ogni tupla come un vettore e assegna lo stesso peso a ogni attributo. Questo metodo di record linkage non è adatto agli attributi categorici, perché è difficile definire la distanza tra due categorie, in particolare se il loro dominio non è ordinato.
  Vengono usati anche altri metodi quando ci sono dataset senza attributi comuni. In questi casi la re-identificazione è più difficile. Un metodo proposto di recente si basa sul clustering. Fondamentalmente, un metodo di clustering viene applicato ai set di dati considerati. Il risultato è un insieme di cluster di tuple e ogni cluster all'interno di un set di dati viene mappato su un cluster all'interno dell'altro set di dati. Tale mappatura viene eseguita utilizzando una funzione di similarità. Sebbene il record linkage sia considerato una minaccia, ci sono molte situazioni in cui può essere utile. Il record linkage può essere utilizzato nella gestione di grandi database per estrarre informazioni importanti sullo stesso soggetto. Ciò è particolarmente utile quando i dati sono distribuiti su server diversi (ad esempio, le informazioni mediche della popolazione sono solitamente distribuite su sistemi diversi e una tecnica di record linkage può essere sfruttata per ricostruire le informazioni associate a un determinato individuo).

Interval Disclosure
La misura di disclosure dell'intervallo viene calcolata in modi diversi, a seconda del tipo di dato dell'attributo (contino o categorico).
Nel caso di attributo categorico, per ogni tupla presente nella table di microdati, gli intervalli classificati (*ranked intervals*) sono costruiti come segue.
Ogni attributo è classificato indipendentemente e un intervallo classificato è definito attorno il valore assunto dall'attributo in ogni tupla t. I ranghi dei valori all'interno dell'intervallo costruito intorno alla tupla t devono differire meno del p% del numero totale di tuple.
Inoltre, il rango al centro dell'intervallo deve corrispondere al valore assunto dall'attributo considerato nella tupla t. Il rischio di divulgazione è quindi la proporzione dei valori originali che cadono nell'intervallo centrato sul valore protetto corrispondente. Se tale proporzione è pari al 100%, un potenziale aggressore(attacker) è sicuro che il valore originale si trovi nell'intervallo attorno al valore protetto.
Nel caso di dati continui, il metodo è simile al precedente. La differenza principale è il modo in cui vengono costruiti gli intervalli classificati: non è possibile sfruttare il ranking e la costruzione si basa sulla deviazione standard dell'attributo.

2.6.2 Perdita di Informazione (Information Loss)

La misura per la perdita di informazione è strettamente correlata allo *scopo* per cui l'informazione verra usata. Poiché gli scopi potrebbero essere differenti a non conosciuti a priori, non è possibile stabilire una misura generale per la perdita di informazione basata sul solo scopo. I metodi usati sono quindi, basati sui concetti di *analiticamente validi* e *analiticamente interessanti*, cui vengono definiti come seguono:

- una tabella di microdati protetta è analiticamente valida se conserva approssimativamente le analisi statistiche (ad esempio, media e covarianza) che possono essere prodotte con i microdati originali;
- una tabella di microdati protetta è analiticamente interessante se contiene un numero suffience di attributi che possono essere validamente analizzati.

Generalmente, ci sono due strategie per calcolare la perdita di informazione:

i) confrontando direttamente le tuple dei microdati protetti con le tuple dei microdati originali.

ii) confrontando le statistiche calcolate sui micordati protetti con le stesse statistiche valutate sui microdati originali.

Descriviamo ora l'idea di base di alcune delle più comuni misure di perdita di informazioni, suddivise in due categorie in base al tipo di dati degli attributi. Sono stati proposti altri metodi, sia per tecniche specifiche di protezione dei

microdati sia per casi generici.

Dati Continui
Per misurare la perdita di informazioni, la statistica di interesse (ad esempio, matrici di co-varianza, matrici di correlazione o loro varianti) viene valutata sia sui dati originali che su quelli protetti e la differenza tra i due valori viene calcolata. Le discrepanze tra le due statistiche possono essere valutate in tre modi diversi:

- *errore quadratico medio*
- *errore medio assoluto*
- *variazione della media*

Oltre alle misure statistiche, i dati possono essere confrontati prima e dopo l'applicazione di una tecnica di protezione dei microdati, calcolando nuovamente la differenza con uno dei tre metodi sopraccitati. È importante notare che il valore della perdita di informazioni deve avere un valore massimo (ad esempio, 100 se si utilizza una notazione percentuale) per confrontare metodi diversi che hanno la stessa scala di calcolo della perdita di informazioni.

Dati Categorici
Le misure di perdita di informazione introdotte brevemente per gli attributi continui non sono direttamente applicabili agli attributi categorici. In questo caso, le misure principali sono tre:

- *confronto diretto*
- *confronto tra tabelle di contigenza*
- *misura di entropia*

Il confronto diretto dei valori degli attributi categoriali richiede la definizione di una funzione di distanza tra le categorie. Nel caso di categorie non ordinate, la distanza tra la categoria c_1 nei microdati originali e la corrispondente categoria c_2 nei microdati protetti è:
$\begin{cases} \text{uguale a 0, se le due categorie sono le stesse} \\ 1, \text{ altrimenti} \end{cases}$

Al contrario, se esiste un ordinamento tra le categorie, la distanza tra le categorie c_1 e c_2 è pari al numero di categorie tra c_1 e c_2 diviso per il numero totale di categorie. La misura di confronto delle tabelle di contingenza consiste nel confrontare le tabelle di contingenza corrispondenti. Una misura basata sull'entropia può essere utilizzata quando una tabella di microdati è stata protetta applicando le tecniche di soppressione locale, global recoding o PRAM. L'idea è che la perdita di informazione possa essere misurata utilizzando l'*entropia di Shannon*, poiché il processo di mascheramento viene modellato come il rumore aggiunto ai microdati originali quando vengono trasmessi attraverso un

canale rumoroso. La misura della perdita di informazione utilizza la probabilità condizionale (la probabilità di un valore nei microdati originali, una volta che il valore nei microdati protetti è dato).

2.6.3 Combnazione di Rischio di Disclosure e Perdita di Informazione

Le tecniche di protezione dei microdati descritte in questo capitolo hanno un impatto diverso sull'utilità dei dati e sul rischio di divulgazione. Per poter valutare tecniche alternative di protezione dei microdati, abbiamo bisogno di un quadro di riferimento per valutare la bontà di una tecnica di protezione.
Il rischio di divulgazione e la perdita di informazioni devono quindi essere combinati. Un metodo semplice consiste nel calcolare la media dei due valori e scegliere la tecnica (e l'impostazione dei parametri) con il punteggio più alto. Un altro metodo è quello delle *mappe di confidenzialità R-U*, che è un grafico in cui la misura dell'utilità dei dati (l'inverso della perdita di informazioni) è riportata sull'asse x e il rischio di divulgazione è riportato sull'asse y. Per ogni tecnica di protezione dei microdati, viene tracciata una linea sul piano cartesiano con un punto per ogni parametro impostato. Sulla base del grafico ottenuto, è possibile confrontare le varie tecniche di protezione e scegliere la più adatta. Una volta scelta la tecnica di protezione, le mappe di riservatezza R-U possono essere utilizzate anche per la selezione dei parametri. È importante notare che una mappa R-U è solo un metodo per correlare il rischio di divulgazione e la perdita di informazioni; tali misure devono essere calcolate utilizzando uno dei metodi sopra menzionati.
Un altro approccio per bilanciare l'utilità dei dati e il rischio di divulgazione è rappresentato dal concetto di tabella k-minima con k-anonimity (guarda il capitolo "k-anonimity"). Il k-anonimity stabilisce una soglia limite inferiore di rischio di divulgazione per una tabella, assicurando che ogni tupla della tabella non possa essere correlata a meno di k rispondenti. L'approccio alla k-anonimità mira a trovare (applicando tecniche di generalizzazione e soppressione) una tabella k-minima, cioè una tabella che non generalizza più di quanto sia necessario per raggiungere la soglia k. In altre parole, una tabella k-minima è una tabella che minimizza la perdita di informazioni.

Le misure descritte dovrebbero essere utilizzate prima che i dati vengano rilasciati per verificare se la protezione è adeguata alle "richieste di riservatezza dei respondent e alle esigenze di informazione dei destinatari dei dati".
Dopo l'applicazione delle tecniche di protezione , i micordati protetti possono essere controllati e rilasciati solo se sono conformi ad un certo grado di protezione. Queste misure possono essere usate anche dal destinatario dei dati per valutare la "protezione dei respondent e l'utilità dei dati"

2.7 Conclusioni

Oggi le società globali interconnesse hanno una grande richiesta sulla dissem-inazione e la condivisione di informazioni. Mentre nel passato le informazioni rilasciate erano principalmente in forma tabulare e statistica, molte situazioni oggi chiamano per il rilascio dei microdati specifici. Per affrontare questo prob-lema sono state proposte numerose tecniche di protezione e, in questo capitolo, sono state descritte le tecniche di base per la protezione contro il disclosure dei microdati, classificando queste in tecniche di mascheramento e tecniche di generazione di dati sintentici. Per riassumere, le tecniche di generazione di dati sintentici proteggono i dati trasformando i loro valori. D'altro canto le tecniche di generazione di dati sintentici proteggono i dati sostituendoli con nuovi dati i quali conservano le proprietà statistiche originali. Inoltre sono state illustrate anche le principali misure adottate, solitamente, per valutare la riservatezza dei dati e l'utilità dei microdati protetti.

Chapter 3

Cloud Privacy Issues

Garantire la sicurezza significa garantire $\mathbb{C}$onfidenzialità, $\mathbb{I}$ntegrità e $\mathbb{A}$ccesso (=Disponibilità) ai dati. Nello scenario cloud tre sono i momenti chiave:

- memorizzazione dei dati
- gestione dei dati
- processazione dei dati

3.1 Introduzione

Usare il cloud è conveniente per benefici come ad esempio la scalabilità e l'elasticità, però comporta anche dei risbolti problematici perchè il proprietario dei dati non ne ha più il pieno controllo, con conseguente minaccia per la sicurezza che può minare la fiducia nell'adozione del cloud computing. Il NIST distingue 4 modelli di sviluppo del Cloud:

1. *privato*: mantenuto su una rete privata
2. *pubblico*: una organizzazione offre servizi cloud a terzi
3. *commmunity*: infrastruttura condivisa da diverse compagnie ma con interessi comuni
4. *ibrido*: un cloud composto da una combinazione di cloud che siano 1,2 oppure 3

Sempre lo stesso NIST identifica 3 modelli di servizio:

1. *IaaS*: **Infrastructure** as a Service
2. *PaaS*: **Platform** as a Service
3. *SaaS*: **Software** as a Service

3.2 $\mathbb{CIA}$ nel Cloud

I problemi di sicurezza possono essere classificati con il paradigma $\mathbb{CIA}$. $\mathbb{C}$ è garantire riservatezza delle informazioni salvate esternamente, $\mathbb{I}$ è garantire l'autenticità dei dati, $\mathbb{A}$ richiede che il provider soddisfi dei livelli di servizio attesi.
In uno scenario complesso si richiede l'esecuzione di query sui dati e comunque operazioni in cui entra in gioco anche un rapporto di fiducia con il cloud provider (o fornitore) che può essere **completamente affidabile** (assunto in caso di cloud privati), **curioso** (archiviazione ed elaborazione coinvolgono informazioni sensibili), **pigro** (ovvero potenzialmente non affidabile nel caso in cui debba gestire info sensibili), **dannoso** o bizantino (fornitore si comporta in modo improprio nella gestione, archiviazione ed elaborazione dei dati compromettendo $\mathbb{CIA}$)

3.3 Problemi e sfide

Non vi è una ricetta come soluzione per preservare $\mathbb{CIA}$, però nonostante i differenti aspetti si possono considerare diversi scenari:

3.3.1 Protezione dei dati al rilascio

Un problema dei dati rilasciati nel cloud è garantire la loro protezione (ovvero rispettare la CIA); solitamente gli utenti devono fidarsi completamente dei cloud provider. Questi garantiscono riservatezza da esterni, ma non vi è nessuno che garantisca che il proprietario del cloud possa accedere ai miei dati personali salvati.

- Una possibile soluzione potrebbe essere allora la **cifratura dei dati prima** di rilasciarli ai cloud provider. La cifratura quindi garantisce $\mathbb{C}$ e $\mathbb{I}$. Si preferisce la critto simmetrica. Ma usare la critto presenta problemi quando si vuole fare un *recupero fine dei dati*.
- Si adotta allora la frammentazione, al posto della critto, che protegge da associazioni di dati:
  - dividendo le parti di informazioni
  - salvandole in frammenti separati non collegabili
- Si può optare anche, nella versione *two can keep a secret*, per fare sì che i frammenti separati siano proprio due provider non comunicanti tra loro, ciascuno dei quali detiene memorizzata una porzione di dati.
- Talvolta vengono cifrati solamente gli attributi sensibili oppure non si cifra nulla se vi è fiducia nei confronti del provider.

$\mathbb{A}$ è invece garantita dalla replicazione dei dati su più cloud provider.

3.3.2 Accesso a grana fine ai dati nel cloud

Mantenere i dati cifrati nel cloud impedisce la decifrazione da parte dei proprietari per eseguire query. Per realizzare questo approccio è necessario però operare query su dati cifrati, come ad esempio

- la **crittografia omomorfica**: questa però ha svantaggi come ad es:
  - realizzabilità solo su operazioni basilari
  - elevato costo computazionale
- **CryptDB**: supporta maggiori operazioni e migliora l'efficienza rispetto alla omomorfica. In questo modello ogni relazione è cifrata a livello di colonna con diversi livelli di cifratura, ognuno dei quali supporta l'esecuzione di una specifica operazione SQL. Funzionamento: quando server riceve la query, determina il livello di profondità di cifratura e il proxy invia al provider la chiave di quel livello per rimuovere i livelli soprastanti di cifratura.
- Un altro approccio consiste nell'**allegare indici** ai dati cifrati per recuperare informazioni a grana fine ed eseguire query. L'indicizzazione può essere *diretta* (1:1) oppure *hash* (N:1) oppure *piatta* (N:1). Queste tre tecniche forniscono diverse garanzie di protezione. Quella hash e piatta garantiscono migliore riservatezza.

3.3.3 Accesso selettivo ai dati nel cloud

Idea è che differenti utenti possano avere differenti viste dei dati. Si parla allora della creazione di vere e proprie ACL, spesso però sono delegate al provider (che medierà ogni richiesta di accesso) e questo richiede fiducia da parte del cliente. Se non vi è totale fiducia si può combinare ACL con crittografia con chiavi diverse, a seconda delle autorizzazioni che si detengono all'atto della richiesta di accesso. Una possibile soluzione al problema del cambio di autorizzazioni nelle ACL è dato dalla *sovracrittografia*.
Inoltre, per alleggerire il carico computazionale, si può operare crittografia basata su attributi (ABE), critto a chiave pubblica che regola l'accesso ai dati in base agli attributi descrittivi associati ai dati stessi.

3.3.4 Privacy dell'utente

Idea è di consentire l'accesso ai dati a utenti non registrati nel sistema, cioè senza che dichiarino la propria identità. L'accesso deve essere quindi basato sulle proprietà degli utenti e non sull'identità. Tale accesso può essere basato su:

- attributi
- credenziali
- certificati

3.3.5 Privacy delle query

In alcuni scenari non sono privati solo i dati o gli utenti, ma anche gli accessi che questi ultimi fanno per accedere ai dati. Una soluzione è Private Information Retrieval, che però ha un alto costo computazionale. Idea è prelevare l'i-esimo bit di una stringa senza rivelare al server quale bit si sia prelevato. Una soluzione meno onerosa è **RAM Oblivious**: *ORAM* e *PathORAM*, basati su una struttura dati *gerarchica* e *dinamica* basata su chiave (detta *indice Shuffle*) modifica dinamicamente (shuffling) ad ogni accesso la posizione fisica dei dati, distruggendo la corrispondenza statica dati-blocchi fisici di memorizzazione. Viene creato un B+ albero contiene nodi con blocchi effettivi e blocchi fittizi per creare confusione al provider affinchè non rintracci il blocco mirato. L'indice di Shuffle predispone n ricerche false per non fare capire quale elemento si sia cercato veramente.

3.3.6 Integrità della computazione e delle query

In caso di mancata fiducia con il provider, il cliente deve poter verificare che il risultato di una query sia *corretto* (= risultato calcolato sui dati originari), *completo* (= non mancano dati dal risultato) e *fresco* (= query è stata eseguita sulla versione più recente dei dati).
Esistono soluzioni:

- **deterministiche**: strutture dati autentiche come ad es. schemi di concatenamento delle firme (consentono la verifica di ordinamento tra le tuple → verifica dell'integrità delle query) e Merkle hash tree (organizza i dati in una struttura ad albero basandosi su un attributo con cui si verifica il risultato della query).
- **probabilistiche**: come ad esempio l'inserimento di *tuple false* nei dati, *replica* di dati, *pre-calcolo dei token* associati ai risultati di una query.
  La probabilità di trovare una compromissione dell'integrità dipende dalla quantità di controlli applicati.
  Una possibile soluzione per la valutazione dell'integrità dei join calcolati da un provider non attendibile è usare server di archiviazione per inserire informazioni di controllo

3.3.7 Esecuzione collaborativa delle query con provider multipli

Scenario in cui ci sono più provider e anche più proprietari dei dati. Importante è garantire che le informazioni non vengano rese accessibili, rilasciate o trapelate: occorre un rilascio selettivo dei dati. [...]

3.3.8 SLA e Auditing

SLA è un accordo contrattuale che specifica le performance e disponibilità che un provider deve mantenere. In passato SLA era sinonimo di disponibilità, tempo

di risposta,...) oggi indica anche criteri di sicurezza (crittografia, protezione perimetrale)

3.3.9 Multi-locazione e virtualizzazione

Multi-locazione significa capacità di fornire servizi informatici a diversi utenti usando infrastruttura cloud comune. Ogni utente dell'infrastruttura cloud condivide risorse di calcolo, come memoria e archiviazione... attività che vengono ben coordinate grazie alla virtualizzazione, che offre grande flessibilità ma introduce problemi di sicurezza come ad esempio **colpire** l'hypervisor (= sw che crea e coordina le macchine virtuali), riguardare l'**allocazione/deallocazione** delle risorse. Il rischio è quello di perdere informazioni in modo improprio se le risorse virtuali non sono isolate nel loro assegnamento al singolo utente

3.4 Conclusioni

La rapida crescita di piattaforme cloud sollecita la considerazione di problemi e preoccupazioni riguardanti la sicurezza in questo scenario, soprattutto per quanto riguarda $\mathbb{CIA}$.

Chapter 4

Supporting User Privacy Preferences in Digital Interactions

4.1 Introduzione

Per la disponibilità dei servizi online si verifica la situazione in cui il server che fornisce il servizio e l'utente che lo richiede sono ignoti l'uno all'altro. Di conseguenza sistemi di access control tradizionali non possono essere adottati in quanto non adatti a scenari open.

Soluzioni proposte sono basate su ABAC (Attribute-Based Access Control). Policy che regolamentano l'accesso ai servizi definiscono condizioni che il client richiedente deve soddisfare per avere accesso.

Il server, a seguito di una richiesta di accesso risponde con le condizioni che il client deve soddisfare per essere autorizzato ad accedere al servizio.

Per accedere il client rilascia **certificati digitali** come **credenziali** firmate da una **CA**. L'adozione delle credenziali nell'Access Control ha diversi vantaggi:

- Non serve <*username,password*> per ogni servizio.
- Offre migliore protezione dall'acquisizione impropria dei privilegi di accesso.

L'uso di credenziali è stata approfonditamente studiata dal lato server con definizione di diversi *policy languages*, *policy engines* e strategie per la comunicazione delle consizioni di accesso.

Le soluzioni trovate, sebbene potenti ed espressive, non supportano completamente gli specifici criteri di protezione dei client. Infatti i client necessitano di un sistema di preferenze che permetta di supportare una propria definizione di

sensitivity/privacy per i propri dati. Queste preferenze verranno poi usate nella situazione in cui più di un subset di credenziali soddisfano la Access Control Policy definita dal server.

Questo capitolo mostra una overview delle problematiche di privacy legate ai sistemi open, oltre che alcune soluzioni. Le sezioni riportano:

- Concetti Base e *Desiderata*.
- Soluzione Cost-Sensitive Trust Negotiation.
- Soluzione Point-Based Trust Management.
- Soluzione Logical-Based Minimal Credential Disclosure.
- Soluzione Privacy Preferences in Credential-Based Interactions

4.2 Concetti Base e *Desiderata*

4.2.1 Client Portfolio

Definizione 10 (*Client* Portfolio)
Insieme delle informazioni che un cliente può fornire al server per avere accesso ad un servizio è racchiuso in un ***Portfolio*** *contenente*

- ***Credenziali*** *firmate da third party.*
- ***Dichiarazioni*** *che comprendono proprietà non certificate dichiarate dall'utente.*

Struttura Credenziale

Ogni credenziale c è composta da:

- Identificatore univoco $id(c)$.
- Emittente $issuer(c)$.
- Un set di attributi $attributes(c)$.
- La tipologia della credenziale $type(c)$.

Gerarchia Tipologie

I tipi di credenziale sono organizzati tipicamente in una gerarchia radicata, dove i nodi intermedi sono astrazione delle specifiche credenziali presenti sulle foglie. Un esempio è in fig. 4.1.

La gerarchia è nota sia al client che al server. Il server tuttavia crea richieste di credential types in quanto non può conoscere le effettive credenziali possedute dal client[1].

[1] Inoltre client può avere più credenziali dello stesso tipo

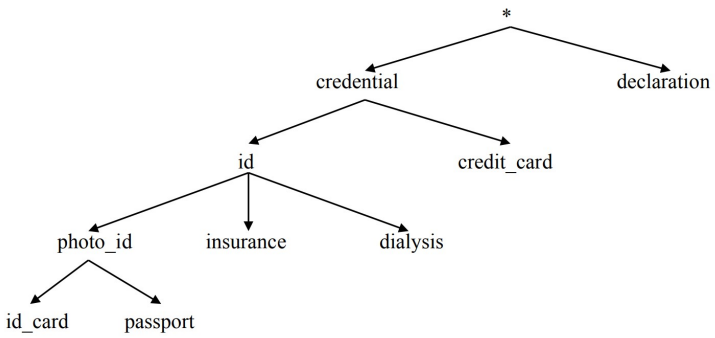


Figure 4.1

4.2.2 Atomicità Credenziali

Le credenziali si distinguono in **atomiche** e **non-atomiche**.

Le credenziali **atomiche** sono la tipologia più comune (vedi X.509) e possono essere rilasciato solo interamente. Per questo anche attributi non richiesti vengono rivelati al server.

Le credenziali **non-atomiche** permettono di limitare il rilascio di attributi rivelando selettivamente un subset degi attributi certificati dalla credenziale.

Ovviamente le *declaration* sono credenziali non-atomiche.

Attributi

Attributi nelle credenziali sono caratterizzati da:

- Tipo
- Nome
- Valore

Essi possono essere *credential-independent* e dipendere dal client ma non dalla credenziale, o *credential-dependent* e dipendere da client e credenziale che la certifica.

4.2.3 Policy di Disclosure

Le policy lat server che regolano l'accesso sono definite sugli attributi e le credenziali fornite dall'utente, questo perchè il server considera solo la gerarchia di credenziali (conosciuta a priori anche dal server).

Una Access Control Policy (policy di controllo di accesso) è quindi una espressione booleana composta da condizioni $term_1\, op\, term_2$, dove *op* è un operatore del tipo $<, >, =$ etc.

Un esempio di policy: *(type(c)= insurance)* $\wedge$ *(c.Company* $\neq$ *'A')*

4.2.4 Trust Negotiation

L'interazione tra client e server mira a stabilire una relazione di fiducia (trust) tra le parti per permettere l'accesso al servizio. Questa interazione si basa tipicamente sul rilascio di credenziali, spesso subordinato al rilascio di altre credenziali da parte della controparte.

Questa sequenza di scambi di credenziali è definita **strategia** e soddisfa le policy di server e client.

Alcune strategie adottate permettono il rilascio di credenziali solo se esiste una *strategia di successo* (successful strategy) che garantisce l'accesso al servizio, mentre altre rilasciano credenziali appena sono richieste.

Il fatto che esistano preferenze sul rilascio di credenziali, sia da parte del server che del client, comporta che le successful strategies non siano tutte equivalenti.

4.2.5 Client Privacy Preferences

Per quanto riportato, è necessario fornire i client di un modello flessibile ed espressivo per rappresentare le preferenze di privacy. Questo modello dovrebbe avere le seguenti caratteristiche:

- **Specific di Preferenze a Grana Fine**: il modello deve supportare la definizione di preferenze per ogni attributo per ogni **istanza** di credenziale.
- **Ereditarietà delle Preferenze di Privacy**: se le istanze di credenziali sono numerose, il modello deve poter applicare sensibilità ai tipi astratti e così a tutte le istanze per cui non era stato specificato un livello di sensibilità.
- **Rel. d'Ordine Parziale e Operatore Composizione**: ci deve essere ordinamento parziale per determinare se una informazione è più o meno sensibile di un'altra. Deve esistere un operatore di composizione $\bigoplus$ che permette di calcolare il valore di sensibilità di un set di attributi/credenziali.
- **Associazioni Sensibili**: rilascio combinato di un set di attr./cred. risulta più o meno sensibile della somma dei rilasci singoli. Il modello deve comprendere anche questa definizione di preferenza.
- **Vincoli di Rilascio**: modello deve permettere di definire vincoli sul rilascio combinoato di parti del portfolio, per evitare o limitare il rilascio dell'associazione tra questi.

- **Context-Based Preferences**: preferenze dipendono dal contesto (rilasci certificato medico a una farmacia, non a un hotel).
- **History-based Preferences**: rilascio dipende da interazioni passate (se a server X ho rilasciato un set, uso sempre quello e non fornisco nuove cred.).
- **Prova di Possesso**: nuove tecnologie permettono di rilasciare prove di possesso di certificati e di soddisfazione requisiti di accesso. Risulta necessario poter specificare preferenze anche per queste prove (proof).
- **User-Friendly Preference Specification**: necessario fornire interfacce ai client, che possono non essere abituati a sistemi di controllo dell'accesso, per poter specificare facilmente le preferenze.

4.2.6 Preferenze Privacy del Server

Mentre la comunicazione della policy server completa favorisce la privacy del client, la comunicazione dei soli attributi coinvolti favorisce la privacy del server.

Ricordare che anche server ha un portfolio di credenziali e attributi oltre alle policy.

Il sistema lato server deve considerare:

- **Disclosure Policy**: server deve poter decidere, a grana fine, come il rilascio policy dovrebbe essere regolato.
- **Comunicazione Policy**: necessario un meccanismo che trasforma la A.C.Policy in modo da proteggere privacy della policy e permettere al client di sapere cosa è richiesto.
- **Integrazione con Meccanismo Client**: Ci deve essere integrazione ra i sistemi lato server e lato client.

4.3 Cost-Sensitive Trust Negotiation

Sensibilità espressa con costo $w(c)$ associata ad ogni credenziale c nel portfolio e con ogni policy di access control p. p viene definita come espressione booleana sulle credenziali della controparte, mentre i valori booleani delle credenziali c sono espressi come *vero* se già rivelati, $false$ altrimenti.

Il costo di sensibilità, definito da utente, modella quanto l'owner ritiene sensibile il rilascio della credenziale e delle informazioni sensibili che certifica.

In fig. 4.2 un esempio di portfolio per client e server.

L'obbiettivo per client e serveer è quindi quello di minimizzare il costo di credenziali e policy scambiate **in una strategia di negoziazione che ha successo**.

$id(c)$	$w(c)$	Policy p regulating c	$w(p)$
MyIdCard	2	TRUE	0
MyPassport	4	TRUE	0
MyCreditCard	10	POS_register	5
MyDialysis	20	pharmacy_register	10
MyInsurance	15	pharmacy_register	10

Table 2 An example of client portfolio and policies regulating its disclosure

$id(c)$	$w(c)$	Policy p regulating c	$w(p)$
MyPOSRegister	2	TRUE	0
MyPharmacyRegister	5	passport ∨ id_card	4

Table 3 An example of server portfolio and policies regulating its disclosure

Figure 4.2: Due esempi di portfolio con policy di rilascio

4.3.1 Minimizzare il Costo

Il problema della minimizzazione del costo viene formulata come segue:

Definizione 11 (Problema del Minimo Costo di Sensibilità)
Dati:

- C_s *set di credenziali e servizi server;*
- P_s *set di policy server;*
- C_c *set di credenziali client;*
- P_c *set policy client;*
- $w : C_s \cup P_s \cup C_c \cup P_c \rightarrow \mathbb{R}$ *funzione costo;*
- $s \in Cs$ *il servizio richiesto.*

Per minimizzare il costo in una successful strategy occorre trovare una sequenza di scambio tale che:

1. *s è rilasciato al client;*
2. *ogni policy è soddisfatta prima del rilascio della corrispettiva credenziale;*
3. *la somma dei costi è* ***minima****.*

Il problema è NP-difficile e con costo esponenziale nel numero di input(credenziali e policy). Tuttavia due soluzioni euristiche (non ottimali) ottengono una buona soluzione con costo polinomiale. Queste considerano o meno il costo delle policy e sono brevemente[2] descritte qui di seguito:

[2]La completa descrizione degli algoritmi penso esuli dallo scope di questo riassunto

Senza Considerare Costo delle Policy

Le policy non sono associate ad un costo, quindi possono essere pubblicate liberamente.

Viene definito il DAG **policy graph** $G(V, A, w)$ che modella le policy di rilascio delle credenziali sia lato server che lato client.

Un esempio del grafo è riportato in fig.4.3 che segue:

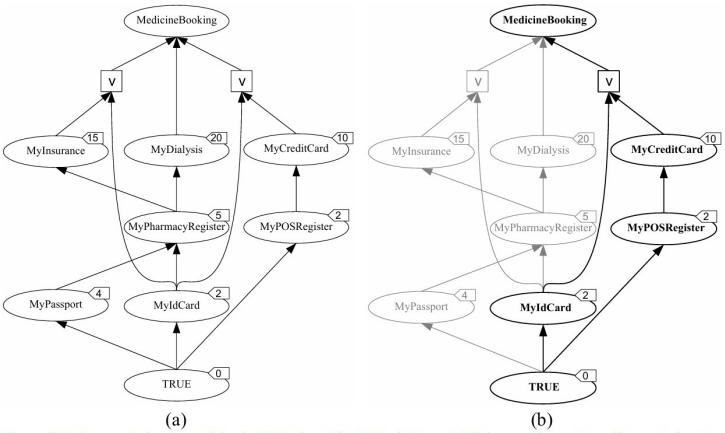


Figure 2 Policy graph for the policies in Table 2 and in Table 3 (a), and Minimum Directed Acyclic graph for the *MedicineBooking* service (b)

Figure 4.3: Policy graph e suo DAG minimo.

Il primo step riguarda sempre la disclosure delle policy, e ciò permette di costruire il grafo.

Formalmente il problema si riduce pertanto al calcolo del DAG minimo.

Considerando il Costo delle Policy

Soluzione greedy che si divide in due step:

1. Le parti adottano una *eager strategy* che consiste nello scambiarsi i nomi delle credenziali e i relativi costi, previa soddisfazione della relativa policy, in modo da identificare una strategia che abbia successo.

2. Applicazione della strategy trovata.

In fig.4.4 viene mostrato uno scambio per i portfolio mostrati precedentemente:

Problematiche Aperte

Limitazioni dei due approcci:

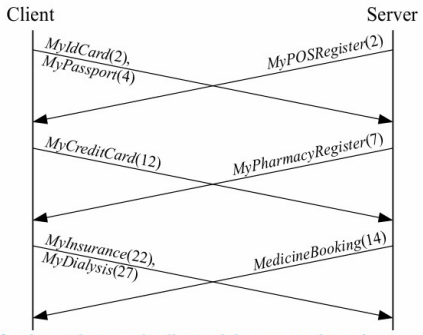


Figure 3 Sequence of exchanges between the client and the server to determine a successful negotiation strategy

Figure 4.4: Scambio di credenziali tra le parti

- mancano le restrizioni sul rilascio delle policy;
- le parti non collaborano a minimizzare il costo a scapito della propria privacy (interessi delle parti);
- non supporta completamente la definizione desiderata espressa precedentemente.

4.4 Point-Based Trust Management

Questo sistema cambia il modello riguardante l'accesso alle risorse. Viene associato un punteggio `ptr` ad ogni credenziale (in base al livello di affidabilità) ed una soglia `thr` di punti richiesti per ottenere accesso ad ogni servizio.

Il client deve quindi presentare un subset delle sue credenziali la cui somma dei punteggi supera la soglia richiesta dal server.

Allo stesso modo il client valuta il rilascio di ogni credenziale nel suo portfolio con un privacy score `ps`. Più è alto il ps, maggiore è la sensibilità della credenziale per il client[3].

Infine, siccome le policy possono essere sensibili, il server non rilascia `thr` o il punteggio `pt` che associa alle credenziali, così come il client non rilascia i `ps`. Non conoscendo cosa è richiesto nasce quindi il *Credential Selection Problem*.

4.4.1 Credential Selection Problem

Il problema consiste nel:

- Trovare un subset di credenziali che soddisfa `thr`.
- Trovare un subset che sia minimo nella somma dei valori di `ps`.

E un problema NP-difficile.

[3] Cient rilascia credenziali con valori di `ps` minori.

4.4.2 Dynamic Programming Algorithm

Approccio che riscrive il problema in un *knapsack problem*: vengono inseriti nel knapsack le credenziali che non verranno rilasciate. Il knapsack ha capacità data dalla somma dei `pt` - il valore di soglia `thr`

Tramite una matrice di appoggio vengono calcolate tutte le possibili modalità di riempimento del knapsack un elemento alla volta: ogni cella contiene il valore totale del knapsack per quello step di riempimento.

Tuttavia `thr` e i `pt` vengono quindi comunicati. Questo non è accettato nello scenario considerato, al che gli autori hanno proposto una soluzione di programmazione dinamica che permette alle parti di risolvere insieme il problema knapsack tramite crittografia homomorphica.

Problematiche Aperte

- Comunicazione set di credenziali su cui basare lo scambio.
- `thr` non basta, server necessita di policy più espressive.
- Focus è sulla negoziazione e non sul management privacy preferences.

4.5 Logical-Based Minimal Credential Disclosure

Può non essere facile per un utente finale definire le preferenze con valori numerici come nelle soluzioni precedenti. Inoltre valori numerici potrebbero imporre accidentalmente gerarchie come side effects.

Vengono quindi introdotti valori qualitativi per i valori di preferenza. Le credenziali sono considerate singleton (certificano un solo attributo) e le policy server sono considerate note a tutti.

Utente deve infine scegliere una *strategy* tra diverse opzioni, filtrate da tutte le strategie possibili tramite le preferenze utente.

4.5.1 Preferenze Qualitative

La soluzione pratica sfrutta la composizione di Pareto[4].

In sintesi, si rappresentano tutte le strategie di successo in una table che riporta il rilascio di ogni attributo.

Le preferenze sono invece espresse da foreste. Nodi a seguire di u arco direzionato sono più sensibili dei nodi alla partenza dell'arco.

Le strategie sono comparate attraverso la comparazione degli attributi secondo le relazioni di preferenza espresse.

[4] Qui vengono volutamente omessi i dettagli del funzionamento

Problematiche Aperte

- Richiede intervento utente per la scelta finale.
- Non considera le credenziali atomiche.

4.6 Privacy Preferences in Credential-Based Interactions

Soluzione che rispetta le *desiderata* creando un portfolio modeling che permette di:

- rappresentare credenziali atomiche e non;
- rappresentare dichiarazioni e attributi distinguendo chiaramente tra credential-dependent e credential-independent.

Risulta più facile associare preferenze ad ogni credenziale e attributo.

4.6.1 Rappresentazione Grafica del Portfolio

Il portfolio viene rappresentato come un grafo bipartito $G(V_C \cup V_A, E_{CA})$ quale quello in figura fig.4.5.

Il grafo ha a sinistra vertici rappresentanti le credenziali, a destra attributi, mentre gli archi associano gli attributi alle credenziali che li contengono. Viene inoltre specificata l'atomicità delle credenziali con il simbolo nero accanto al vertice credenziale.

Sensitivity Labels

Utente specifica preferenze a granularità fine andando ad applicare una *sensitivity label* ad ogni credenziale e attributo.

Il modello inoltre permette di definire:

- Viste sensibili: rilascio combinato porta più informazione della composizione, quindi il rilascio combinato è modellato e porta una label di sensibilità positiva (addizionale).
- Dipendenze: rilascio combinato porta meno informazione della composizione, il rilascio combinato ha label negativa.
- Viste proibite: subset di componenti del portfolio che non possono essere rilasciati insieme (proibito).
- Limiti al rilascio (Disclosure Limitations): in un subset posso rilasciare al massimo n elementi insieme.

La label corrispondente ad un rilascio risulta corrispondente all'applicazione di una composizione ($\bigoplus$) sulle label considerate.

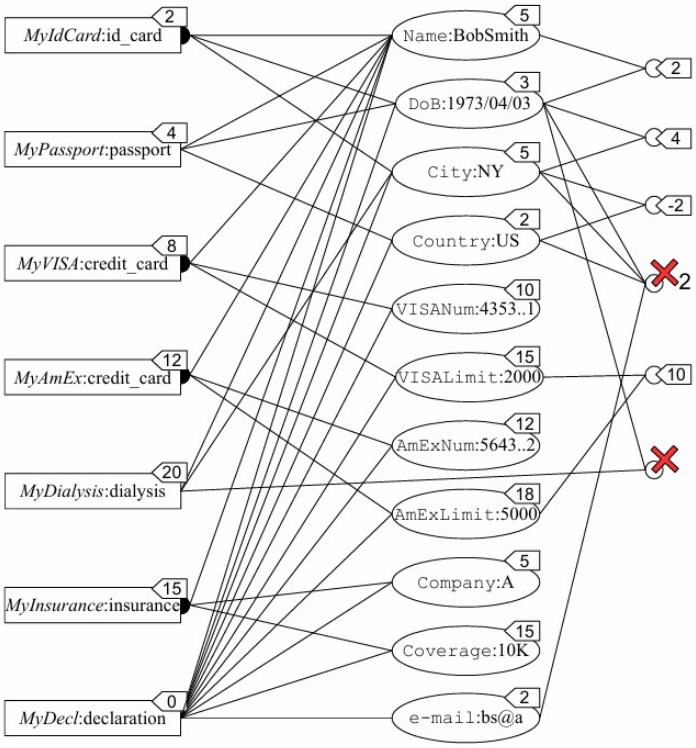


Figure 4.5: Rappresentazione gafica del portfolio utente

4.6.2 Disclosure Valide e Disclosure Minime

Non tutti i subset del portfolio rappresentano rilasci validi per ottenere un servizio.

Tre condizioni regolano un corretto rilascio:

- Certificabilità: ogni attributo è associato ad una credenziale di cui, rilasciando l'attributo, rilascio l'esistenza.
- Atomicità: ogni attributo in credenziale atomica comporta rilascio di tutti gli attributi nell'associazione.
- Esposizione dell'associazione: se tutti gli attributi legati ad una associazione sono rilasciati, conto anche l'associazione nel calcolo $\bigoplus$.

Data una policy server è necessario determinare una disclosure minima, problema NP-difficile.

Esistono sia una soluzione euristica basata sui grafi sia una soluzione modellata sul problema Max-SAT, utilizzando quindi risolutori per Max-SAT.

4.6.3 Context e History

Sono state proposte estensioni per gestire restrizioni basate sia sul contesto che sulla storia dei rilasci.

4.6.4 Problematiche Aperte

- Label potrebbero non essere facili da esprimere per gli utenti, per cui sarebbe meglio definire ordinamento parziale.

4.7 Disclosure a Grana Fine di Policy di Accesso Sensibili

NON VISTO A LEZIONE - QUI OMESSO

4.8 Problematiche Aperte Riguardo La Specifica di Preferenze Utente

- Ereditarietà Preferenze: associare preferenza al tipo di credenziale e non alle credenziali specifiche.
- Prova di Possesso: tecnologia permette di mostrare prova di avere credenziale senza rilasciare la credenziale stessa.
- Conoscenza Condivisa: gestione delle credenziali secondo un modello che non garantisce che le tipologie di credenziali siano conosciute da entrambe le parti.
- Integrazione: soluzioni prendono una parte, serve soluzione che protegga server e client contemporaneamente.
- Definire un modello che combini i vantaggi dei modelli proposti senza portare gli svantaggi correlati nelle solucioni attuali.

Chapter 5

Accesso selettivo e privato ai Data center Outsourced

5.1 Introduzione

La crescente quantità di informazioni generate, raccolte, condivise e diffuse al giorno d'oggi rende sempre più difficile ed economicamente costosa la gestione interna dei data center da parte di aziende private e pubbliche. L'ampia disponibilità di fornitori di cloud che offrono servizi di alta qualità per l'archiviazione e la gestione dei dati è quindi una motivazione che spinge le aziende a spostare sempre più spesso i propri data center nel cloud. Sebbene questa tendenza presenti chiari vantaggi economici, introduce anche nuovi problemi di sicurezza. Infatti, quando si sposta un data center nel cloud, i dati non sono più sotto il controllo diretto del proprietario che deve affidarsi a un sistema esterno per fornire le stesse garanzie della gestione interna (ad esempio, disponibilità dei dati, protezione da attacchi esterni, accesso selettivo ai dati, gestione della tolleranza ai guasti). Tuttavia, essendo terze parti esterne, si presume spesso che i fornitori di cloud siano onesti ma curiosi, quindi affidabili nel gestire correttamente i dati che memorizzano, ma non nell'accedere ai loro contenuti. Questa situazione solleva diverse preoccupazioni, soprattutto per quanto riguarda l'adeguata protezione della riservatezza dei dati. Una soluzione efficace consiste nel crittografare i dati prima di esternalizzarli, in modo che le parti non autorizzate (compreso il cloud provider), non conoscendo la chiave di crittografia, non possano accedere al contenuto dei dati in chiaro. La crittografia dei dati prima dell'outsourcing presenta tuttavia alcuni svantaggi.

In primo luogo, pur nascondendo efficacemente i dati in chiaro agli occhi del fornitore, la crittografia di tutti i dati con un'unica chiave richiederebbe che tutti gli utenti abbiano una visibilità completa delle risorse della raccolta dati, oppure che il data owner medi le richieste di accesso ai dati per imporre un accesso selettivo. In secondo luogo, la crittografia complica la valutazione delle

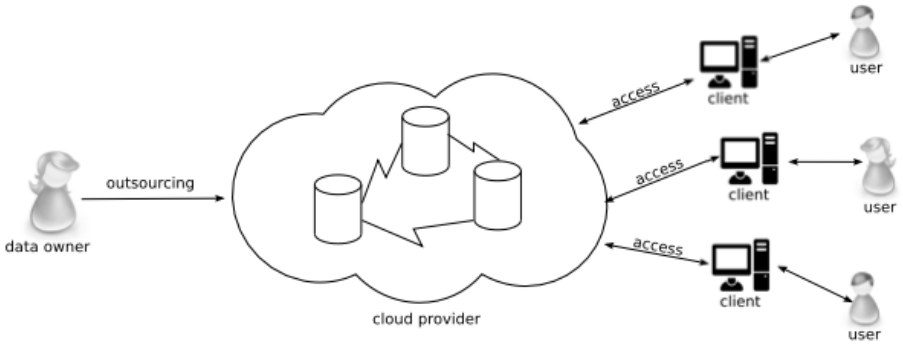
query, poiché il cloud provider non può valutare direttamente le query degli utenti sui dati crittografati. In terzo luogo, nei casi in cui anche le query poste dagli utenti devono essere protette, la crittografia potrebbe non fornire sufficienti garanzie di protezione. Per superare questi problemi, sono state proposte diverse tecniche che mirano a supportare un accesso selettivo e privato ai dati esternalizzati. Queste tecniche si basano sull'uso di una crittografia selettiva, nel senso che i diversi pezzi di dati vengono crittografati con chiavi diverse a seconda di chi può accedervi. Gli indici sono invece utilizzati dai cloud provider per selezionare i dati da restituire in risposta a una query, eventualmente anche senza rivelare l'obiettivo della query stessa. Sebbene, singolarmente, queste tecniche rappresentino soluzioni efficaci, l'adozione combinata di crittografia selettiva e indici può causare violazioni della riservatezza che devono essere affrontate con attenzione.

In questo capitolo presentiamo una panoramica delle tecniche proposte per consentire ai dati di auto-applicare la politica di controllo degli accessi definita dal loro proprietario e per supportare la valutazione delle query sui dati criptati. La Figura 1 illustra lo scenario di riferimento in cui il data owner affida i propri dati a un fornitore di cloud e gli utenti accedono a tali dati. Il resto del capitolo è organizzato come segue:

- La sezione 2 mostra come i dati criptati possano imporre restrizioni al controllo degli accessi, senza richiedere l'intervento del data owner o la collaborazione del server di archiviazione.
- La sezione 3 presenta una panoramica delle tecniche proposte per supportare la valutazione delle query su dati criptati.
- La sezione 4 descrive nuove soluzioni per accedere a raccolte di dati in outsourcing senza rivelare al server di archiviazione l'obiettivo dell'interrogazione.
- La sezione 5 illustra i problemi di privacy che sorgono quando si combinano soluzioni per l'applicazione del controllo degli accessi con tecniche di indicizzazione e introduce soluzioni preliminari a questo problema.
- La Sezione 6 presenta le nostre osservazioni conclusive.

5.2 Access Control Enforcement

Le informazioni memorizzate nei data center possono essere di qualsiasi tipo: database relazionali, documenti XML, file multimediali, ecc. Per semplicità, ma senza perdita di generalità, in questo capitolo assumiamo che i dati memorizzati nel cloud siano organizzati in un database relazionale, con la nota che tutti gli approcci illustrati nel seguito possono essere facilmente adattati per operare su qualsiasi modellazione logica dei dati.



Quindi consideriamo una relazione r definita sullo schema $R(a_1, ..., a_n)$, dove l'attributo a_i è definito sul dominio $D_i, i = 1, ..., n$. Al server che ha i dati, la relazione r è rappresentata come una relazione r^k, definita sullo schema $R^k(tid, enc)$, con tid la chiave primaria numerica aggiunta alla relazione crittata e enc la tupla crittata.
Ogni tupla t in r è rappresentata come una tupla crittata t^k in r^k, dove $t^k[tid]$ è scelto randomicamente dal data owner, e $t^k[enc] = E_k(t)$, con E una funzione di crittazione simmetrica con chiave k

Sono state proposte diverse tecniche per applicare il controllo degli accessi senza l'intervento né del server di archiviazione, per motivi di riservatezza, né del data owner, per motivi di efficienza. Queste soluzioni si basano sull'idea che i dati si *auto-applichino(self-enforce)* alle restrizioni di accesso selettivo attraverso la crittografia, come illustrato nel seguito di questa sezione.

5.2.1 Selective Encryption

Una soluzione promettente per imporre il controllo degli accessi ai dati in outsourcing si basa sulla crittografia selettiva, che adotta chiavi di crittografia diverse per tuple diverse e distribuisce selettivamente le chiavi agli utenti autorizzati. Ogni utente può decifrare e quindi accedere a un sottoinsieme di tuple, a seconda delle chiavi che conosce. La politica di autorizzazione che regola quale utente può leggere quale tupla è definita dal data owner prima di esternalizzare la relazione r

La politica di autorizzazione può essere rappresentata come una matrice binaria di accesso M con una riga per ogni utente u e una colonna per ogni tupla t, dove:

$$\begin{cases} M[u,t] = 1 \text{ se u può accedere a t;} \\ M[u,t] = 0 \text{ altrimenti.} \end{cases}$$

per illustrare, considera la relazione PATIENTS nella tabella sottostante:

PATIENTS

	SSN	Name	ZIP	MarStatus	Illness
t_1	123456789	Ann	22010	single	gastritis
t_2	234567891	Barbara	24027	divorced	neuralgia
t_3	345678912	Carl	22010	married	gastritis
t_4	456789123	Daniel	20100	married	gastritis
t_5	567891234	Emma	21048	single	neuralgia
t_6	678912345	Fred	23013	married	hypertension
t_7	789123456	Gary	22010	widow	gastritis
t_8	891234567	Harry	24027	widow	hypertension

Fig. 2 An example of relation

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
A	1	1	0	1	1	1	1	0
B	1	1	1	1	1	0	0	0
C	1	1	1	0	1	1	0	0
D	0	0	0	1	1	1	0	1
E	0	0	0	1	1	1	0	0

Fig. 3 An example of access matrix

La Figura 3 illustra un esempio di matrice di accesso che regola l'accesso alle tuple della relazione PATIENTS da parte degli utenti A, B, C, D ed E. La j_{esima} colonna della matrice rappresenta la lista di controllo degli accessi $acl(t_j)$ della tupla t_j, per ogni $j = 1, ..., |r|$. Ad esempio, con riferimento alla matrice di Figura 3, $acl(t_1) = ABC$.
La politica di cifratura, che definisce e regola l'insieme delle chiavi utilizzate per cifrare le tuple e la distribuzione delle chiavi agli utenti, deve essere equivalente alla politica di autorizzazione, nel senso che ogni utente deve poter decifrare tutte e solo le tuple a cui è autorizzato ad accedere. Le soluzioni che traducono una politica di autorizzazione in una politica di crittografia equivalente hanno due principali desiderata progettuali:

i) garantire che ogni utente debba gestire una sola chiave;

ii) crittografare ogni tupla con una sola chiave (cioè, nessuna tupla viene replicata).

Per soddisfare questi due requisiti, gli approcci di crittografia selettiva si basano su tecniche di derivazione delle chiavi che permettono di calcolare una chiave di

crittografia k_j a partire dalla conoscenza di un'altra chiave k_i e (eventualmente) di un'informazione disponibile al pubblico.
Per determinare quale chiave può essere derivata da quale altra chiave, le tecniche di derivazione delle chiavi richiedono la definizione preliminare di *key derivation hierarchy*. Una gerarchia di derivazione delle chiavi può essere rappresentata graficamente come un grafo diretto con un vertice v_i per ogni chiave k_i del sistema e un arco (v_i, v_j) dalla chiave k_i alla chiave k_j *sse* k_j può essere derivata direttamente da k_i.

Si noti che la derivazione delle chiavi può essere applicata a catena, nel senso che la chiave k_j può essere calcolata a partire dalla chiave k_i se esiste un percorso (di lunghezza arbitraria) da v_i a v_j nella gerarchia di derivazione delle chiavi. Una gerarchia di derivazione delle chiavi può avere forme diverse, come descritto di seguito.

- *Chain of vertices*: la chiave k_i associata al vertice v_j è calcolata applicando la funzione *one-way* alla chiave k_i del predecessore nella catena. Nessuna informazione pubblica è necessaria
- *Tree hierarchy*: la chiave k_i associata al vertice v_j è calcolata applicando la funzione *one-way* alla chiave k_i del suo antenato diretto, e una label pubblica l_j associata con k_j. Le label pubbliche sono necessarie per garantire che i diversi figli di uno stesso nodo dell'albero abbiano chiavi diverse.
- *DAG hierarchy*: le chiavi nella gerarchia possono avere più di un antenato diretto e ogni arco della gerarchia è associato a un token pubblicamente disponibile. Date due chiavi k_i e k_j , e l'etichetta pubblica l_j di k_j , il token $t_{i,j}$ permette di calcolare k_j a partire da k_i e l_j . Il token $t_{i,j}$ è calcolato come $t_{i,j} = k_j \oplus (k_i, l_j)$, dove $\oplus$ è l'operatore XOR bitwise e f è una funzione crittografica deterministica. Tramite $t_{i,j}$, tutti gli utenti che conoscono (o sono in grado di ricavare) la chiave k_i possono ricavare anche la chiave k_j

5.3 BEL e SEL

In caso di modifiche alla politica di autorizzazione, la politica di crittografia deve essere aggiornata di conseguenza, per garantire la loro equivalenza. Poiché la chiave utilizzata per criptare ogni tupla t in r dipende dall'insieme di utenti che possono accedervi, potrebbe essere necessario criptare nuovamente le tuple coinvolte nell'aggiornamento della politica con una chiave diversa che solo gli utenti delle nuove liste di controllo degli accessi conoscono o possono ricavare. Un approccio banale per imporre un'operazione di grant/revoke sulla tupla t richiede che il data owner:

i) scarichi t_k dal server;

ii) la decifri;

iii) aggiorni la gerarchia di derivazione delle chiavi se non include un vertice che rappresenta il nuovo insieme di utenti in $acl(t)$;

iv) cripti t con la chiave k' associata al vertice che rappresenta $acl(t)$;

v) eventualmente aggiorni il catalogo pubblico contenente i token.

Ad esempio, si consideri la politica di crittografia delle figura sottostante:

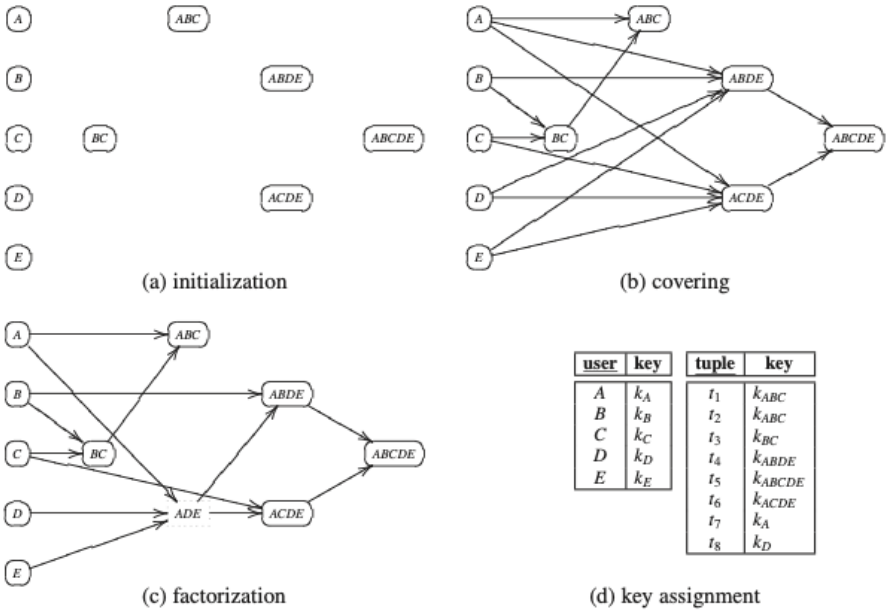


Fig. 5 Definition of an encryption policy equivalent to the access control policy in Figure 3

e si supponga che all'utente D sia concesso l'accesso alla tupla t_1. Il data owner dovrebbe scaricare t_1^k; decifrarla usando la chiave k_{ABC}; inserire un vertice che rappresenti $acl(t_1) = ABCD$ nella gerarchia di derivazione delle chiavi; criptare t_1 con k_{ABCD}; e caricare la tupla criptata sul server, insieme ai token necessari agli utenti A, B, C e D per derivare k_{ABCD}. Questo approccio, pur essendo efficace e in grado di imporre correttamente gli aggiornamenti dell'autorizzazione, lascia al data owner l'onere di gestire l'aggiornamento. Inoltre, le operazioni di ricrittografia sono computazionalmente costose. Per limitare l'overhead del data owner, gli autori propongono di utilizzare due livelli di crittografia (ciascuno caratterizzato da una propria politica di crittografia) per delegare parzialmente al server la gestione delle operazioni di grant e revoke.

- Il *Base Encryption Layer(BEL)* viene applicato dal data owner prima di esternalizzare il set di dati. La gerarchia di derivazione delle chiavi BEL viene costruita in base alla politica di autorizzazione esistente al momento

dell'inizializzazione. In caso di aggiornamento della politica, la BEL viene aggiornata solo inserendo eventualmente dei token nel catalogo pubblico (ovvero dei bordi nella gerarchia di derivazione delle chiavi BEL).
Si noti che ogni vertice v nella gerarchia di derivazione delle chiavi BEL ha due chiavi: una chiave di derivazione k (utilizzata solo per la derivazione delle chiavi) e una chiave di accesso k_a (utilizzata per crittografare le tuple, ma che non può essere sfruttata ai fini della derivazione delle chiavi).

- Il *Il Surface Encryption Layer(SEL)*viene applicato dal server sulle tuple che sono già state crittografate dal data owner a BEL. Il server applica dinamicamente gli aggiornamenti dei criteri di autorizzazione, eventualmente cifrando nuovamente le tuple e modificando la gerarchia di derivazione delle chiavi del SEL per riflettere correttamente gli aggiornamenti. A differenza di BEL, i vertici della gerarchia di derivazione delle chiavi di SEL sono associati a una singola chiave k_s

Intuitivamente, con l'approccio di over-encryption, un utente può accedere a una tupla t solo se conosce le chiavi utilizzate per crittografare t in BEL e SEL. Al momento dell'inizializzazione, le politiche di crittografia di BEL e SEL coincidono, ma cambiano immediatamente e diventano diverse a ogni aggiornamento della politica. Le operazioni di grant e revoke sono applicate come segue:

- *Grant*: Quando l'utente u ottiene l'accesso alla tupla t, deve conoscere la chiave utilizzata per crittografare t sia in BEL che in SEL. Pertanto, il proprietario dei dati aggiunge un token nella gerarchia di derivazione delle chiavi di BEL dal vertice che rappresenta u al vertice la cui chiave è stata utilizzata per crittografare t (cioè il vertice che rappresenta $acl(t)$ al momento dell'inizializzazione). Il proprietario chiede quindi al server di aggiornare la gerarchia di derivazione delle chiavi in SEL ed eventualmente di criptare nuovamente le tuple. La tupla t deve infatti essere cifrata al SEL con la chiave del vertice che rappresenta $acl(t) \cup \{u\}$ (che viene eventualmente inserito nella gerarchia). Oltre a t, anche altre tuple possono aver bisogno di essere ricifrate al SEL per garantire la corretta applicazione dell'aggiornamento della politica. Infatti, le tuple criptate con la stessa chiave di t a BEL e che l'utente u non può leggere devono essere criptate a SEL con una chiave che u non conosce (e non può ricavare). Il proprietario dei dati deve quindi assicurarsi che ogni tupla t_i che condivide la chiave di cifratura BEL con t sia cifrata in SEL con la chiave del vertice che rappresenta $acl(t_i)$. Ad esempio, si consideri la matrice di accesso della Figura 3 e le politiche di crittografia di BEL e SEL che la applicano. BEL e SEL che la applicano nella Figura 6, e supponiamo che all'utente D sia concesso l'accesso alla tupla t_1. La Figura 7 illustra le politiche di crittografia di BEL e SEL dopo l'applicazione dell'operazione di grant.

  Per applicare questa modifica alla politica di controllo degli accessi, il proprietario dei dati deve innanzitutto aggiungere un token che consenta all'utente D di derivare la chiave di accesso del vertice $ABC(k^a_{ABC})$ utiliz-

zata per crittografare t_1 a BEL (bordo tratteggiato nella figura). Inoltre, chiederà al server di aggiornare la gerarchia di derivazione delle chiavi SEL per aggiungere un vertice che rappresenti $ABCD$. La tupla t_1 viene quindi sovracrittografata in SEL con la chiave di questo nuovo vertice.

- *Revoke*: Quando l'utente u perde il privilegio di accedere alla tupla t, il proprietario dei dati chiede semplicemente al server di criptare nuovamente (a SEL) la tupla con la chiave associata all'insieme $acl(t)\backslash\{u\}$ di utenti. Se il vertice che rappresenta questo insieme di utenti non è rappresentato nella gerarchia di derivazione delle chiavi di SEL, il server aggiorna prima la gerarchia inserendo il nuovo vertice e poi cripta nuovamente la tupla. Ad esempio, si considerino le politiche di cifratura di BEL e SEL nella Figura 7 e si supponga che il proprietario dei dati revochi a B il privilegio di accedere a t_4. Il proprietario dei dati richiede al server di modificare SEL (BEL non è influenzato dalle operazioni di revoke) per garantire che la tupla t_4 sia crittografata con una chiave che l'utente B non può ricavare. A tal fine, t_4 viene nuovamente crittografata con la chiave k^s_{ADE} . La Figura 8 illustra le politiche di crittografia di BEL e SEL dopo l'applicazione dell'operazione di revoke. Si noti che il vertice $ABDE$ è stato rimosso dalla gerarchia poiché non è necessario per l'applicazione della politica né utile per ridurre il numero di token.

Poiché la gestione delle operazioni di (ri)crittografia in SEL è delegata al server, esiste il rischio di collusioni con gli utenti. Infatti, combinando le loro conoscenze, un utente e il server possono decifrare tuple a cui né il server né l'utente possono accedere. Ad esempio, con riferimento alla politica di crittografia della Figura 8, il server e l'utente D possono accedere alla tupla t_2 combinando le loro conoscenze. Infatti, questa tupla è cifrata con la chiave di accesso k^a_{ABC} a BEL, nota all'utente D in quanto utilizzata per cifrare t_1, e con la chiave k^s_{ABC} a SEL, nota al server. La collusione rappresenta un rischio per la corretta applicazione della politica di autorizzazione, ma questo rischio è limitato. Infatti, la collusione tra un utente u e il server permette di decifrare una tupla t a cui non si è autorizzati ad accedere solo se a u viene concesso il privilegio di leggere una tupla $t' \neq t$ (diversa da t) che è cifrata con la stessa chiave di t a BEL. Infatti, u conosce la chiave con cui t è cifrato a BEL (in quanto necessaria per accedere a t') mentre il server conosce la chiave con cui è cifrato a SEL (in quanto gestisce tutte le chiavi di cifratura a SEL).
Il rischio di collisione può quindi essere mitigato al prezzo di utilizzare un numero maggiore di chiavi a BEL, cioè utilizzando la stessa chiave di crittografia a BEL solo per le tuple le cui acl hanno la probabilità di evolvere nello stesso modo.

5.3.1 Scrivere privilegi

La soluzione descritta nella sezione precedente, pur imponendo efficacemente i privilegi di lettura e gli aggiornamenti, presuppone che la relazione in outsourc-

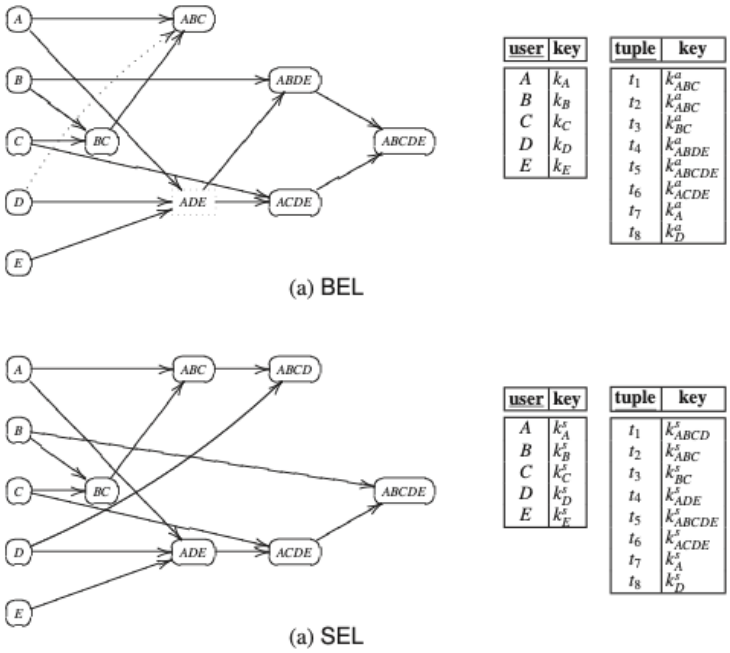


Fig. 8 Encryption policies at BEL and SEL in Figure 7 after revoking B access to t_4

ing sia di sola lettura (cioè che solo il proprietario possa modificare le tuple). Per consentire al proprietario dei dati di autorizzare selettivamente altri utenti ad aggiornare i dati esternalizzati, questo approccio è stato integrato con una tecnica specifica per gestire i privilegi di scrittura. L'approccio di [11] associa a ogni tupla un write tags (cioè un valore casuale indipendente dal contenuto della tupla) definito dal proprietario dei dati. L'accesso ai write tags è regolato attraverso una crittografia selettiva: il *write tag* della tupla t è crittografato con una chiave nota solo agli utenti autorizzati a scrivere t (cioè gli utenti specificati nella sua lista di accesso alla scrittura, indicata con $acl_w(t)$) e dal server. In questo modo, solo il server e gli autori autorizzati hanno accesso al write tags in chiaro di ogni tupla. Il server accetterà quindi una richiesta di scrittura su una tupla quando l'utente richiedente dimostrerà di conoscere il write tags corrispondente.

Poiché la chiave utilizzata per crittografare il write tags di una tupla deve essere condivisa tra il server e gli autori di tuple, è necessario estendere la gerarchia di derivazione delle chiavi con il server di archiviazione. Tuttavia, il server non può accedere alle tuple esternalizzate in chiaro e quindi non può essere trattato

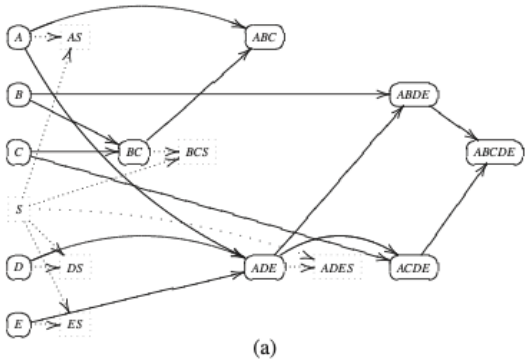
come un ulteriore utente autorizzato (cioè con la possibilità di derivare le chiavi nella gerarchia).
Le chiavi utilizzate per criptare i write tags sono quindi definite in modo tale che:

i) gli utenti autorizzati possano calcolarle applicando una funzione di hash sicura a una chiave che già conoscono (o che possono ricavare tramite una sequenza di token);

ii) il server possa ricavarle direttamente da una chiave k_S assegnatagli, tramite un token appositamente aggiunto alla gerarchia di derivazione delle chiavi.

Si noti che le chiavi utilizzate per crittografare i write tags non possono essere utilizzate per derivare altre chiavi nella gerarchia. Ad esempio, si consideri la politica di crittografia delle figure 5(c-d) e si assuma che:
$acl_w(t_1) = acl_w(t_7) = A, acl_w(t_2) = acl_w(t_3) = BC, acl_w(t_4) = ADE, acl_w(t_5) = acl_w(t_8) = Deacl_w(t_6) = E$.

La Figura 9(a) illustra la gerarchia di derivazione delle chiavi, ampliata con la chiave kS assegnata al server S e le chiavi necessarie per crittografare i write tags (i vertici e gli spigoli aggiuntivi sono tratteggiati nella figura). Le figure 9(b-c) riassumono le chiavi assegnate agli utenti e al server e le chiavi utilizzate per crittografare le tuple della relazione PAZIENTI e i loro write tags, rispettivamente.



user	key
A	k_A
B	k_B
C	k_C
D	k_D
E	k_E
S	k_S

tuple	read_key	write_key
t_1	k_{ABC}	k_{AS}
t_2	k_{ABC}	k_{BCS}
t_3	k_{BC}	k_{BCS}
t_4	k_{ABDE}	k_{ADES}
t_5	k_{ABCDE}	k_{DS}
t_6	k_{ACDE}	k_{ES}
t_7	k_A	k_{AS}
t_8	k_D	k_{DS}

Fig. 9 Encryption policy in Figures 5(c-d) extended to the enforcement of write privileges

L'approccio dell'over-ecnryption, pur essendo efficace per imporre gli aggiornamenti di una politica di autorizzazione alla lettura, non può purtroppo essere adottato per imporre la concessione e la revoca delle autorizzazioni alla scrittura. Un possibile metodo per imporre privilegi di scrittura dinamici funziona come segue.

- *Grant*: Quando all'utente u viene concesso il privilegio di modificare la tupla t, il write tag viene crittografato con una chiave nota al server e agli

utenti in $acl_w(t) \cup \{u\}$. Se la gerarchia di derivazione delle chiavi non la include, tale chiave viene creata e aggiunta correttamente alla gerarchia. Ad esempio, con riferimento alla politica di crittografia della Figura 9, si supponga che all'utente B sia concesso il privilegio di scrittura su t_4. Il write tag della tupla deve essere crittografato con la chiave k_{ABDES}, che viene inserita nella gerarchia di derivazione delle chiavi, mentre la chiave k_{ADES} può essere rimossa.

- *Revoke*: Quando all'utente u viene revocato il privilegio di scrittura sulla tupla t, è necessario definire un nuovo write tag per t, con un valore indipendente dal tag precedente (ad esempio, può essere scelto adottando una funzione casuale sicura). Questo è necessario per garantire che u, che non è ignaro, non possa sfruttare la sua conoscenza del precedente write tag della tupla t per eseguire operazioni di scrittura non autorizzate. Dopo che il tag è stato generato, viene crittografato con una chiave nota al server e agli utenti in $acl_w(t) \backslash \{u\}$. Ad esempio, con riferimento alla politica di crittografia della Figura 9, si supponga che all'utente C venga revocato il privilegio di scrittura su t_3. Il write tag della tupla deve essere modificato e crittografato con la chiave k_{BS}, che deve essere inserita nella gerarchia di derivazione delle chiavi.