Protezione e Integrità dei Dati nel Cloud

Parte V

Indice

Capitolo 1

Encryption

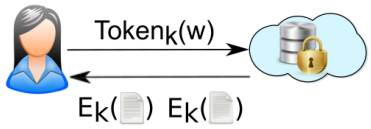
Il *server* potrebbe essere ***honest-but-curious***, non dovrebbe avere accesso alle risorse; voglio garantire confidenzialità anche rispetto a lui.

Un modo per ottenerla è utilizzare l'*encyption*: si aggiunge un livello di protezione attorno ai dati sensibili che li rende non leggibili a chi non è autorizzato.

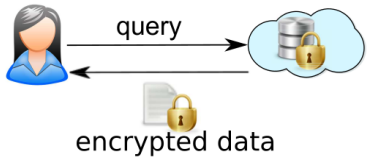
Di base voglio avere una criptazione dei dati; il problema è il **bilanciamento tra protezione e funzionalità**, ovvero sulle *query* che è possibile fare sui dati.

Approcci per accesso a diversi livelli di granularità

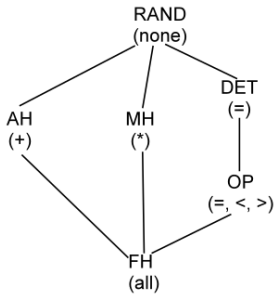
- ***Keyword-based searching:*** passo un *token* già criptato che viene usato per fare ricerca sui dati criptati (voglio trovare dove c'è una certa parola/espressione booleana)



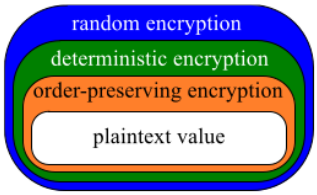
- **Crittografia omomorfica:** crittografia che supporta le operazioni direttamente sul cifrato

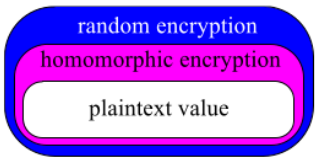


- ***Encryption Schemas:*** ogni colonna può essere cifrata con un diverso schema crittografico (*random, add homomorphic, deterministic, order preserving, . . .*)



- ***Onion Encryption:*** cifro i dati con diversi livelli *a cipolla*, ognuno dei quali supporta l'esecuzione di una specifica *query SQL*; l'idea è che *scopro il dato solo quando mi serve*





- **Indicizzazione:** associo degli indici ai metadati Nella seconda tabella:

Accounts

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

Accounts$^{k}_{I}$

Counter	Etuple	I_A	I_C	I_B
1	x4Z3tfX2ShOSM	π	α	μ
2	mNHg1oC010p8w	ϖ	α	κ
3	WsIaCvfyF1Dxw	ξ	β	η
4	JpO8eLTVgwV1E	ρ	γ	κ
5	qctG6XnFNDTQc	ς	δ	θ
6	4QbqCeq3hxZHklU	ι	ε	κ

nella seconda colonna c'è la tupla criptata; nelle ultime tre ci sono gli attributi; si possono avere diversi tipi di indicizzazione:

  - ***Direct*** $(1:1)$

+ riesco a fare query precise

- soggetto ad attacchi di frequenza

Patients

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

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WsIaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

– ***Bucket*** $(n : 1) \rightarrow$ indicizzazione con collisione; ho diversi valori che sono **mappati allo stesso indice**

+ non ho più attacchi di frequenze

+ supporta query di uguaglianza (*se un valore è uguale ad un altro*)

- i risultati avranno delle tuple spurie

- è ancora possibile fare qualche leakage *In questo caso sono comunque*

Patients

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

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WsIaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

esposto perché asma ha 3 occorrenze, dunque sarà per forza associata ad α

– ***Flattened*** $(1 : n) \rightarrow$ ciascun indice deve avere lo stesso numero di occorrenze; significa che i valori che hanno più occorrenze sono associati ad indici diversi

+ rimuovo la possibilità di fare attacchi di inferenze

- sono esposto ad osservazioni dinamiche (magari certi dati sono sempre cercati assieme)

Patients

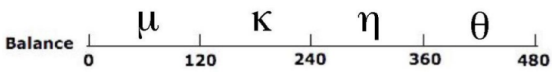
SSN	Name	Illness	Doctor
123…89	Alice	Asthma	Angel
234…91	Bob	Asthma	Angel
345…12	Carol	Asthma	Bell
456…23	David	Bronchitis	Clark
567…34	Eva	Gastritis	Dan
232…11	Eva	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WsIaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

– ***Partition-based:***

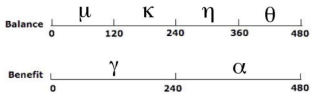
1. si partiziona il dominio di un attributo
2. a ciascuna partizione si assegna un'etichetta
3. il valore in chiaro viene sostituito dall'etichetta



Supporta *query* dove le condizioni sono espressioni booleane del tipo:
- *Attribute* `op` *Value*
- *Attribute* `op` *Attribute*

dove `op`$= \{=, <, >, \leq, \geq\}$

Example



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

Esecuzione delle query:

Ogni query Q sul DB in chiaro viene tradotta in:

1. una query Q_s da eseguire sul server $\rightarrow$ query sull'indice per ottenere le tuple criptate
2. una query Q_c da eseguire sul client $\rightarrow$ decriptare il risultato della query precedente e filtrare le tuple spurie

La traduzione dovrebbe essere fatta in modo tale che il server sia responsabile della maggior parte del lavoro.

Accounts

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

Accounts$^{\mathbf{k}}_{\mathbf{2}}$

Counter	Etuple	I_A	I_C	I_B
1	x4Z3tfX2ShOSM	π	α	μ
2	mNHg1oC010p8w	ϖ	α	κ
3	WslaCvfyF1Dxw	ξ	δ	θ
4	JpO8eLTVgwV1E	ρ	α	κ
5	qctG6XnFNDTQc	ς	β	κ
6	4QbqC3hxZHklU	ι	β	κ

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

– ***Hash-based:*** basate sul concetto di *one-way hash function*; ogni attributo viene mappato ad un indice utilizzando una funzione di hash sicura.

Dat una funzione h e il dominio degli attributi D_i, diciamo che h è **sicura** se:

1. $\forall x, y \in D_i \implies h(x) = h(y)$ (**determinismo**)
2. dati due valori $x, y \in D_i$ tali che $x \neq y$, potremmo avere che $h(x) = h(y)$ (**collisione**, per proteggermi da attacchi di frequenza)
3. la distanza dei valori in chiaro deve essere **indipendente** dalla distanza dei valori di hash (*strong mixing*)

Questo metodo supporta *query* dove le condizioni sono espressioni booleane del tipo:

* $Attribute = Value$
* $Attribute_1 = Attribute_2$, se sono indicizzati con la stessa funzione di hash

La traduzione funziona come nel metodo *partion-based*; non sono supportate *query di range*.

Interval-based queries

- Le tecniche di indicizzazione che preservano l'ordine supportano query di range, ma sono esposte ad inferenza
- Le tecniche di incizzazione che *non* preservano l'ordine non sono esposte ad inferenza, ma non supportano query di range

$\rightarrow$ viene calcolato un $B_+ - tree$ dal client, ed ogni nodo viene criptato come un tutt'uno; successivamente per rispondere alle query l'albero viene visitato (in ambiente trusted).

1.1 *Searchable Encryption*

1.1.1 *Order preserving encryption*

- ***Order Preserving Encryption Schema (OPES):*** prende in input una distribuzione target di valori per gli indici ed applica una trasformazione che preserva l'ordine e rispecchia la distribuzione di input.

  \+ la comparazione può essere fatta direttamente sui dati criptati

  \+ le query non producono tuple spurie

  \- vulnerabile ad attacchi di inferenza

- ***Order Preserving Encryption with Splitting and Scaling (OPESS):***

  Questo schema crea degli indici in modo tale che la loro distribuzione delle frequenze sia piatta.

1.1.2 *Fully homomorphic encryption*

- Permette una performante computazione specifica sui dati criptati
- Decriptando il risultato, si ottiene lo stesso risultato delle stesse operazioni sui dati in chiaro

1.2 Esposizione all'inferenza