

Protezione e Integrità dei Dati nel Cloud

Parte V

Indice

1 Encryption

2

Capitolo 1

Encryption

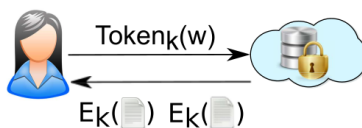
Il *server* potrebbe essere *honest-but-curious*, non dovrebbe avere accesso alle risorse; voglio garantire confidenzialità anche rispetto a lui.

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

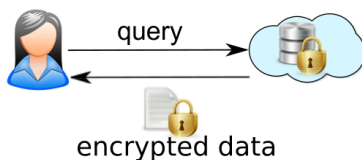
Di base voglio avere una criptazione dei dati; il problema è il **bilanciamento tra protezione e funzionalità**, ovvero sulle *query* che è possibile fare sui dati.

Approcci per accesso a diversi livelli di granularità

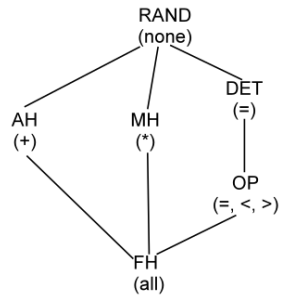
- **Keyword-based searching:** passo un *token* già criptato che viene usato per fare ricerca sui dati criptati (voglio trovare dove c'è una certa parola/espressione booleana)



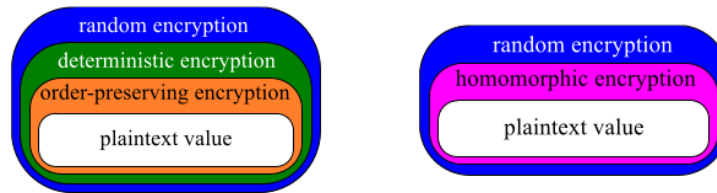
- **Crittografia omomorfica:** crittografia che supporta le operazioni direttamente sul cifrato



- **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 ₁ ^k				
Counter	Etuple	I _A	I _C	I _B
1	x4Z3tfX2ShOSM	π	α	μ
2	mNHg1oC010p8w	ϖ	α	κ
3	WslaCvfyF1Dxw	ξ	β	η
4	JpO8eLTVgwV1E	ρ	γ	κ
5	qctG6XnFNDTQc	ς	δ	θ
6	4QbqCeq3hxZHkIU	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				Patients ^k					
SSN	Name	Illness	Doctor	Tid	Etuple	I _S	I _N	I _I	I _D
123...89	Alice	Asthma	Angel	1	x4Z3tfX2ShOSM	π	κ	α	δ
234...91	Bob	Asthma	Angel	2	mNHg1oC010p8w	ϖ	ω	α	δ
345...12	Carol	Asthma	Bell	3	WslaCvfyF1Dxw	ξ	λ	α	ν
456...23	David	Bronchitis	Clark	4	JpO8eLTVgwV1E	ρ	υ	β	γ
567...34	Eva	Gastritis	Dan	5	qctG6XnFNDTQc	ι	μ	α	σ
232...11	Eva	Stroke	Ellis	6	kotG8XnFNDTaW	χ	\omicron	β	ψ

- **Bucket** ($n : 1$) → 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				Patients ^k					
SSN	Name	Illness	Doctor	Tid	Etuple	I _S	I _N	I _I	I _D
123...89	Alice	Asthma	Angel	1	x4Z3tfX2ShOSM	π	κ	α	δ
234...91	Bob	Asthma	Angel	2	mNHg1oC010p8w	ϖ	ω	α	δ
345...12	Carol	Asthma	Bell	3	WslaCvfyF1Dxw	ξ	λ	α	ν
456...23	David	Bronchitis	Clark	4	JpO8eLTVgwV1E	ρ	υ	β	γ
567...34	Eva	Gastritis	Dan	5	qctG6XnFNDTQc	ι	μ	α	σ
232...11	Eva	Stroke	Ellis	6	kotG8XnFNDTaW	χ	\omicron	β	ψ

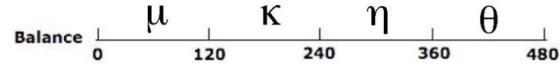
esposto perché asma ha 3 occorrenze, dunque sarà per forza associata ad α

- **Flattened** ($1 : n$) → 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				Patients ^k					
SSN	Name	Illness	Doctor	Tid	Etuple	I _S	I _N	I _I	I _D
123...89	Alice	Asthma	Angel	1	x4Z3tfX2ShOSM	π	κ	α	δ
234...91	Bob	Asthma	Angel	2	mNHg1oC010p8w	ϖ	ω	α	δ
345...12	Carol	Asthma	Bell	3	WslaCvfyF1Dxw	ξ	λ	α	ν
456...23	David	Bronchitis	Clark	4	JpO8eLTVgwV1E	ρ	υ	β	γ
567...34	Eva	Gastritis	Dan	5	qctG6XnFNDTQc	ι	μ	α	σ
232...11	Eva	Stroke	Ellis	6	kotG8XnFNDTaW	χ	\omicron	β	ψ

– **Partition-based:**

- si partiziona il dominio di un attributo
- a ciascuna partizione si assegna un'etichetta
- il valore in chiaro viene sostituito dall'etichetta



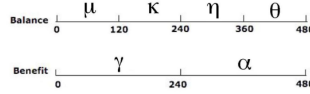
Supporta *query* dove le condizioni sono espressioni booleane del tipo:

- *Attribute* **op** *Value*

- *Attribute* **op** *Attribute*

dove **op** = {=, <, >, ≤, ≥}

Example



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

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			Accounts ₂				
Account	Customer	Balance	Counter	Etuple	I _A	I _C	I _B
Acc1	Alice	100	1	x4Z3tfX2ShOSM	π	α	μ
Acc2	Alice	200	2	mNHg1oC010p8w	ϖ	α	κ
Acc3	Bob	300	3	WslaCvfyF1Dxw	ξ	δ	θ
Acc4	Chris	200	4	JpO8eLTVgwV1E	ρ	α	κ
Acc5	Donna	400	5	qctG6XnFNDTQc	ζ	β	κ
Acc6	Elvis	200	6	4QbqC3hxZHKIU	ι	β	κ

Original query on Accounts	Translation over Accounts ₂ ^k
<pre>Q := SELECT * FROM Accounts WHERE Balance=200</pre>	<pre>Q_s :=SELECT Etuple FROM Accounts₂^k WHERE I_B=κ</pre> <pre>Q_c :=SELECT* FROM Decrypt(Q_s, Key) WHERE Balance=200</pre>

– *Hash-based:*