

## Preferenze Privacy degli Utenti

### Parte IV

# Indice

<b>1</b>	<b>Introduzione</b>	<b>2</b>
1.1	Rilascio diretto . . . . .	3
1.2	Controllo di accesso interattivo . . . . .	4
1.2.1	Interazione senza condizioni da parte del client . . . . .	4
1.2.2	Negoziazione <i>multi-step</i> . . . . .	4
1.2.3	Interazione a due step . . . . .	5
1.3	Preferenze degli utenti . . . . .	6
<b>2</b>	<b>Cost-sensitive Trust Negotiation</b>	<b>7</b>
<b>3</b>	<b>Point-based Trust Management Model</b>	<b>9</b>
<b>4</b>	<b>Logic-based Minimal Credential Disclosure</b>	<b>11</b>
<b>5</b>	<b>Privacy Preferences in Credential-based Interactions</b>	<b>14</b>

# Capitolo 1

## Introduzione

### Privacy dell'identità degli utenti

Gli utenti preferiscono restare anonimi o comunque non condividere troppe informazioni quando operano nel cloud. Alcuni scenari:

- **Tecniche di comunicazione anonima**
- **Privacy in location-based services** (protezione della location quando sensibile)
- **Attribute-based control access:** è un problema lato server, non ci si basa più su chi un utente sia (l'identità) ma sugli attributi che ha (certificati che l'utente presenta)
- **Supporto alle preferenze privacy degli utenti:** problema lato utente; *se mi viene chiesto un documento d'identità, non è che do al server tutto il portafoglio*

Gli utenti potrebbero voler specificare le proprie scelte in termini di politiche del trattamento dei dati, quando:

- condividono delle proprie risorse con server esterni (ad esempio i social media)
- vengono rilasciate informazioni nelle interazioni digitali (ad esempio lascio la carta di credito per accedere a un servizio)

→ Due aspetti di **protezione:**

- **rilascio diretto:** regola quando, a chi e perchè un utente rilascia informazioni (es. sto comprando qualcosa)
- **uso secondario:** regola l'uso e la profilazione dei dati da terze parti; anche questo deve essere sotto il controllo dell'utente

## 1.1 Rilascio diretto

La community di ricerca ha sviluppato diverse tecniche per regolare le interazioni tra *parti sconosciute*, definendo dei meccanismi di **attribute-based access control**: consistono in una dipendenza dell'accesso rispetto alle proprietà che un utente ha. Quello che gli utenti possono fare dipende dagli attributi che possiedono, verificati attraverso i **certificati**.

L'*access control* non risponde più sì o no, ma risponde con i requisiti che il richiedente deve soddisfare per avere l'accesso. Non solo i server vanno protetti ma anche gli utenti, per questo vanno introdotte delle *forme di negoziazione*.

### Esempio

Se vogliamo cambiare filosofia, in un sistema aperto (non so chi è l'utente) se voglio chiedere *"tu soddisfi i requisiti per ottenere l'accesso?"*, nascono una serie di problematiche:

- come specificare l'autorizzazione
- *engine* per il controllo della politica
- anche la politica potrebbe essere confidenziale (*non voglio dirti che faccio certi controlli*)
- come chiedere le cose all'utente
- l'utente può avere delle controrichieste (*hai la certificazione per chiedermi la carta di credito? la cripti?*)

Questo dialogo deve terminare, deve essere **corretto** e **minimale** nei termini delle informazioni rilasciate; tipicamente vengono usati linguaggi basati sul paradigma logico.

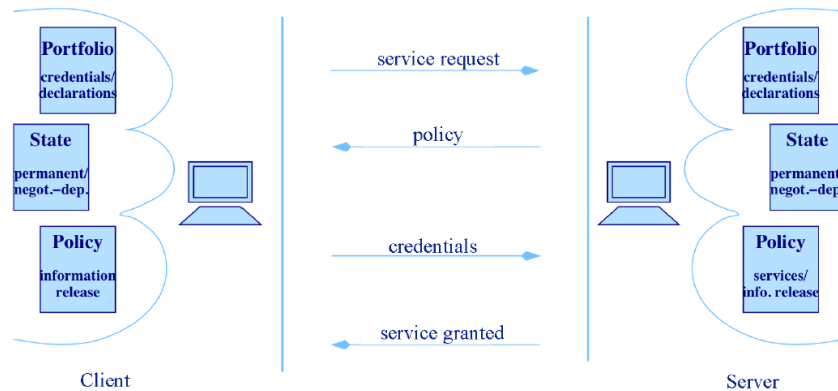
## 1.2 Controllo di accesso interattivo

Il client è colui che richiede il servizio (utente), ha con sé:

- **portfolio** (credenziali e proprietà)
- **stato** (stato di informazioni che vuole mantenere)
- **politica**

Lo stesso vale per il server, cioè colui che offre il servizio.

### 1.2.1 Interazione senza condizioni da parte del client



La policy del server sta ad indicare ciò che il client deve dimostrare, tramite i certificati, per poter accedere al servizio.

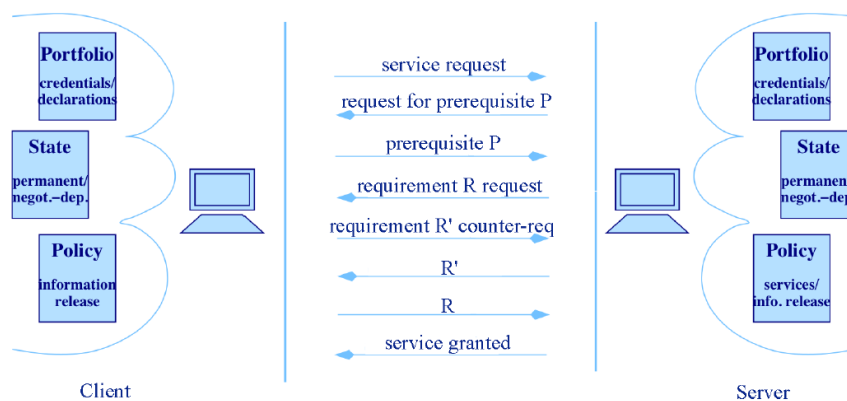
### 1.2.2 Negoziazione *multi-step*

In questo caso c'è una negoziazione tra client e server → bisogna stabilire fiducia tra le due parti.

Il server per essere *privacy-friendly* dovrebbe chiedere i dati tutti assieme.

### 1.2.3 Interazione a due step

Per essere *gentili* con l'utente viene fatta una distinzione tra i prerequisiti per l'accesso (necessari ma non sufficienti) e il requisito vero e proprio con eventuale controrichiesta da parte dell'utente.



### Esistenti/emegenti tecnologie di supporto a ABAC

- U-Prove/Idemix: fornisce avanzate tecnologie di gestione dei certificati (i certificati odierni ti permettono di estrapolare dal certificato solo l'informazione che voglio fornire all'altra parte, senza fornire tutto il certificato).
- XACML: standard di oggi per l'interoperabilità delle politiche di controllo degli accessi

## 1.3 Preferenze degli utenti

Le specifiche di controllo degli accessi non sempre si adattano bene con il problema lato utente:

- **+** sono espressive, potenti e permettono all'utente di specificare se determinate informazioni possono o non possono essere rilasciate
- **-** non permettono agli utenti di esprimere che preferirebbero rilasciare determinate informazioni piuttosto che altre, nel contesto in cui ne sia data la possibilità

→ È necessario fornire agli utenti strumenti per definire in modo efficace le preferenze sulla privacy riguardo al rilascio delle loro informazioni

### *Desiderata*

- **Context-based preferences:** sono disposto a rilasciare un'informazione solo se mi trovo in un certo contesto (*lascio la carta solo quando devo pagare*)
- **Forbidden disclosures:** certe cose insieme non le rilascio
- **Associazioni sensibili:** associazioni che sono sensibili, perché sono *quasi identifier* o perché non voglio che tu le veda
- **Limited disclosure:** *se mi chiedi di essere maggiorenne, te lo dimostro ma non voglio dirti la mia età*
- **Instance-based preferences:** *se la mia carta sta per scadere, preferisco lasciarti quella*
- **History-based preferences:** magari ho già rilasciato qualcosa in passato
- **Proof-based preferences:** *preferisco darti la prova che possiedo un passaporto invece che il passaporto stesso*
- **Non-linkability preferences:** *preferisco lasciarti informazioni che, linkate con terze parti, mi identificano di meno*

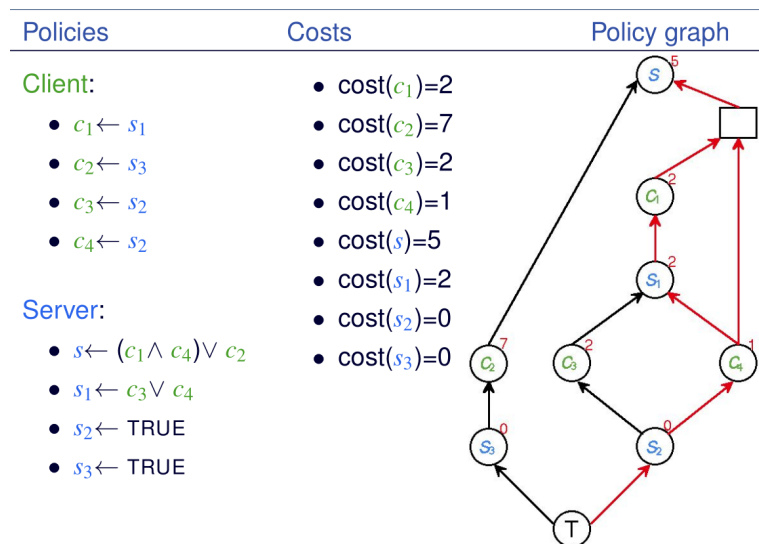
Esistono diversi approcci per regolare la preferenza sulla privacy per gli utenti, che andiamo a vedere nei prossimi capitoli.

## Capitolo 2

# Cost-sensitive Trust Negotiation

- Due parti (client e server) interagiscono per stabilire fiducia reciproca tramite lo scambio di credenziali → *trust negotiation protocol*
- Il **rilascio di una credenziale** è regolato da una politica che specifica dei prerequisiti da soddisfare affinché la credenziale possa essere rilasciata
- Credenziali e politiche sono associate ad un costo  
→ più sono sensibili, più il costo è maggiore

L'obiettivo è di **minimizzare i costi totali** di rilascio di credenziali e politiche durante la *trust negotiation*.





Con  $c_1 \leftarrow s_1$  si intende *io rilascio se tu hai*. Il rettangolo nel *policy graph* corrisponde ad un AND.

### Conclusioni

- La *cost-sensitive trust negotiation* offre un meccanismo per il rilascio delle credenziali in base alla loro sensibilità. Si concentra sulla negoziazione stessa piuttosto che sul controllo da parte dell'utente.
- Supporta soltanto specifiche sulle credenziali; *sensitive association* e *forbidden releases* non possono essere specificate.
- Questo tipo di approccio (minimizzazione del costo totale) ha un'applicabilità limitata; inoltre, la combinazione lineare dei costi potrebbe non essere desiderabile.

## Capitolo 3

# Point-based Trust Management Model

- Il server assegna dei **punti** a ciascuna credenziale
    - rappresentano il livello di affidabilità del proprietario
    - devono essere tenuti privati
  - Il server richiede una **soglia minima di punti totali** per offrire accesso alla risorsa; deve essere tenuto privata
  - Il client assegna a ciascuna sua credenziale un **punteggio** (privato), che indica la **sensibilità** della credenziale
- l'obiettivo è trovare un sottoinsieme delle credenziali del client che **soddisfa la soglia** stabilita dal server che ha **valore di privacy minimo** per il client

Threshold of accessing a resource: 10

SERVER

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

CLIENT

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

Client's options:

- SSN [Points: 10; Sensitivity: 100]
- College ID, Credit card [Points: 11; Sensitivity: 60]
- Driver's license, Credit card [Points: 14; Sensitivity: 80]

### Conclusioni

- Il calcolo della soluzione viene ottenuto convertendo il problema convertendolo al *problema dello zaino*, utilizzando un protocollo sicuro che interagisce con entrambe le parti (client e server), in modo che i punteggi privati non vengano rivelati da una parte all'altra.
- Si concentra sulla negoziazione piuttosto che sul controllo da parte dell'utente
- *sensitive association* e *forbidden disclosure* non possono essere specificate
- il client e il server devono concordarsi sull'universo dei possibili tipi di credenziali (potrebbe compromettere la confidenzialità delle politiche del server)

## Capitolo 4

# Logic-based Minimal Credential Disclosure

- Le parti sono coinvolte in una *trust negotiation* nella quale il rilascio di credenziali è regolato da politiche (problema di bilanciare il linguaggio con la difficoltà delle regole)
- Ogni credenziale è un **singolo attributo**
- Combinando le politiche si ottengono diversi *negotiation paths* (set di credenziali per il rilascio) che soddisfano la negoziazione (sia il server che il client possono chiedere qualcosa)

→ Si usa l'approccio logico per permettere agli utenti di specificare le preferenze sulla privacy e selezionare un *negotiation path*.

Alice's policy	On-line book shop's policy	Negotiation paths											
$c_{name} \leftarrow \text{TRUE}$ $c_{bdate} \leftarrow c_{bbb}$ $c_{telephone} \leftarrow c_{bbb}$ $c_{email} \leftarrow c_{bbb}$ $c_{pcode} \leftarrow c_{bbb}$ $c_{id} \leftarrow c_{bbb}$ $c_{passport} \leftarrow c_{bbb}$ $c_{bname} \leftarrow c_{bbb} \wedge c_{osc}$ $c_{baccount} \leftarrow c_{bbb} \wedge c_{osc}$ $c_{credit\_card} \leftarrow c_{bbb} \wedge c_{osc}$ $c_{pin} \leftarrow c_{bbb} \wedge c_{osc}$	$\text{purchase} \leftarrow p_{register} \wedge p_{payment}$ $p_{register} \leftarrow (c_{name} \wedge c_{bdate} \wedge (c_{email} \vee c_{pcode})) \vee c_{id} \vee c_{passport} \vee ((c_{name} \vee c_{email}) \wedge c_{id})$ $p_{payment} \leftarrow (c_{bname} \wedge c_{baccount}) \vee (c_{credit\_card} \wedge c_{pin})$ $c_{bbb} \leftarrow \text{TRUE}$ $c_{osc} \leftarrow \text{TRUE}$												
			name	bdate	telephone	email	pcode	id	passport	bname	baccount	credit_card	pin
		$S_1$	1	1	0	1	0	0	0	1	1	0	0
		$S_2$	1	1	0	1	0	0	0	0	0	1	1
		$S_3$	1	1	0	0	1	0	0	1	1	0	0
		$S_4$	1	1	0	0	1	0	0	0	1	1	1
		$S_5$	0	0	0	0	0	1	0	1	1	0	0
		$S_6$	0	0	0	0	0	1	0	0	0	1	1
		$S_7$	0	0	0	0	0	0	1	1	1	0	0
		$S_8$	0	0	0	0	0	0	1	0	0	1	1
		$S_9$	1	0	0	0	0	1	0	1	1	0	0
		$S_{10}$	1	0	0	0	0	1	0	0	0	1	1
		$S_{11}$	0	0	0	1	0	1	0	1	1	0	0
		$S_{12}$	0	0	0	1	0	1	0	0	0	1	1
Disclosure sets are represented as binary vectors ⇒ 0 means do not disclose; 1 means disclose													

Nella colonna di *Alice's Policy* sappiamo che Alice sarà disposta a rilasciare determinate informazioni a patto che la controparte abbia determinati certificati.

Ciascuna delle righe di *negotiation paths* corrisponde a un possibile rilascio di credenziali al server, con lo scopo di soddisfare *purchase*. Ogni strategia può essere vista come un vettore binario.

- Preferenza di default: non rilasciare una credenziale è meglio che rilasciarla.
- I set di rilascio sono confrontati secondo la **Pareto composition** ( $>_P$ ):  
 $S_i$  domina  $S_j$  se  $S_i$  ha valori migliori o uguali di  $S_j$ , rispettano le preferenze di rilascio delle credenziali ed è strettamente migliore rispetto ad almeno una credenziale (rilascia qualcosa in meno)

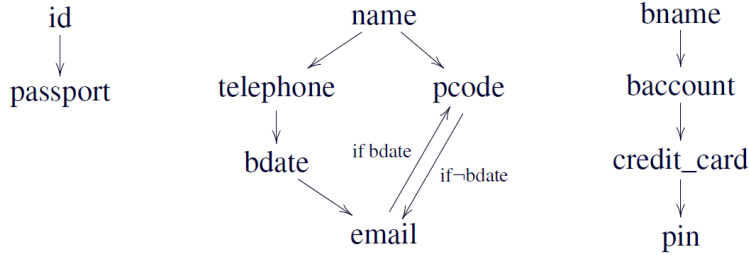
### Example

$S_5$ : [0,0,0,0,0,1,0,1,1,0,0]     $S_9$ : [1,0,0,0,0,1,0,1,1,0,0]

$S_5[i] = S_9[i], i = 2, \dots, 11$  and  $S_5[1] \succ_1 S_9[1]$

$\Rightarrow S_5$  dominates  $S_9$  ( $S_5 \succ_P S_9$ )

La gerarchia specifica le preferenze di rilascio delle credenziali ( $c_i \rightarrow c_j$  significa che preferiamo rilasciare  $c_i$  rispetto a  $c_j$ )



È possibile esprimere preferenze condizionate in relazione a cosa ho rilasciato precedentemente.

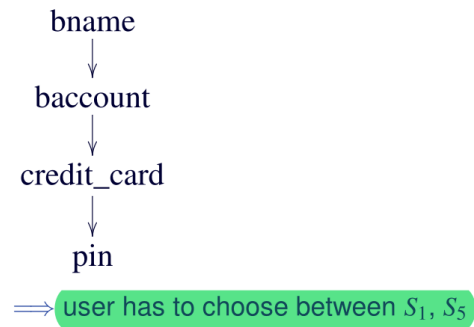
Le righe arancione vengono scartate sulle preferenze degli utenti; le righe rosse sono scartate perché rilasciano più informazioni di quelle necessarie.

### Disclosure sets

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

### Transitive combination of preferences

$S_1$  dominates  $S_2$   
 $S_5$  dominates  $S_6$



### Conclusioni

- Gli utenti sono coinvolti nella scelta del disclosure set
- Vengono assunti solo attributi (non si ragiona in termini di credenziali)
- La specifica di preferenze su gruppi di attributi non è sempre facile
- Credenziali *possession-sensitive* non sono considerate
- *Forbidden releases* non sono supportati

## Capitolo 5

# Privacy Preferences in Credential-based Interactions