**Privacy and Data Protection in Emerging Scenarios**

Gli utenti possono voler rimanere anonimo e non voler revelare troppe informazioni sul loro conto quando sono in rete.

• Anonymous communication techniques (e.g., Mix networks, onion routing, Tor, Crowds)

• Privacy in location-based services

• Attribute-based access control

• Support for user-privacy preferences in information disclosure

È anche possibile che gli utenti vogliano decidere che tipo di policy utilizzare quando comunicano con un server per la rivelazione di informazioni ex: rilaciare informazioni in interazioni in rete acquisto con carta di credito ecc...

2 aspetti di protezione da regolare:

Il direct release: quando un utente si accorda direttamente con il provider del servizio per l’utilizzo delle informazioni rilasciate.

secondary usage: regola il rilascio dell’informazione da parte del provider del servizio ad altre parti.

Attribute based access control: l’accesso alle informazioni non dipende più dalla tua identità ma dalle proprietà che hai, presentazione di certificati per verificare le tue proprietà.

Il controllo non risponde con si o no ma presenta i requisiti necessari per poter accedere al servizio.

È anche possibile che venga introdotta una negoziazione per poter dare delle garanzie all’utente (l’utente vuole garanzie quindi negozia).

**Interactive access-control:**

No condictions by the client: Per quanto ci riguarda qui abbiamo il client e il server, il client vuole fare qualcosa, il server gli dice cosa deve soddisfare, l’utente gli invia le proprietà sotto forma di certificati digitali e il server gli da il servizio.

Multi-step negotiation: Nella seconda versione c’è un rimpallo di negoziazioni (io ti do questo se tu mi dici questo e così via) fino a dare il servizio finale, viene definita anche trust management.

Non si ha la sicurezza sul servizio io fornisco dati ma l’altro non mi da il servizio perché vuole altro

Non so se la policy è necessaria o sufficiente. Suffiente accedo e uso le cose, encessaria mi servono quelle cose solo per presentarmi

Two-step interaction:

Metto in evidenza le condizioni suffienti, ho due step di interazione, nel primo caso metto in evidenza le condizioni necessarie perché la comunicazione possa avvenire e poi metto in evidenza le condizioni sufficienti per poter usufruire del servizio. Questo per essere carini verso l’utente altrimenti uso il multi-step e posso fare qualsiasi cosa

U-prove/Idemix: estraggo l’informazione dalla credenziale, posso presentare la proprietà relativa senza dare la credenziale, esempio ho la data di nascita, non voglio fornirla ma fornisco il fatto che sono maggiorenne. Utilizzo certificati firmati da terze parti.

(XACML) è uno standard per l’interoperabilità di politiche di controllo degli accessi.

Le access control specifications sono un tool potente che però non è sempre il migliore per quanto riguarda il client, ad esempio il client vorrebbe poter accedere dando certi dati piuttosto che altro e non può farlo con quella policy specifica.

Voglio poter dare la possibilità all’utente di fare dei ragionamenti e di decidere quali dati rilasciarmi che mi possano andare bene per il rilascio del servizio, diverse preferenze possono essere dedotte:

Basati sul contesto: (lascio la carta di credito solo se sto facendo un pagamento), non te la

rilascio prima di chiudere il carrello

- Rilasci che non voglio fare: può darsi che io abbia tutte le mie collezioni di proprietà ma

posso averle ciascuna per il proprio contesto (diverse identità digitali che voglio mantenere

scollegate)

- Associazioni sensibili: se sto regolamentando quali proprietà rilascio alla controparte:

l’associazione tra il mio zip code e la mia data di nascita è più sensibile di due informazioni

prese singolarmente.

- Disclosure con limite: se per fare qualcosa devo essere maggiorenne io non voglio dirti

quanti anni ho, ma mi basta dirti che sono maggiorenne.

- Posso avere preferenze basate sul valore delle cose: preferisco rilasciare la mia carta di

credito rispetto al mio account bancario se la mia carta scade tra poco

- Posso avere preferenze basate sulla storia: preferisco rilasciare la mia città se hai già il mio

zip code.

- Rilascio di informazioni con prova: invece di darti il mio passaporto ti dimostro che ho il

passaporto italiano

- Non-linkability preferenze: preferisco rilasciare informazioni che, unite tra loro mi

identificano meno.

- Ecc....

Approcci per realizzare preferenze sulla privacy dell’utente:

**Cost-Sensitive Trust Negotiation**

Trust negotiation, io ti devo dimostrare che ho delle cose e tu mi devi dimostrare che hai delle cose/servizi.

Costo = quanto è sensibile un certo certificato, quanto è sensibile il rilascio.

Ho qualcuno che chiede all’altro un servizio ma ho dei vincoli rispetto alla trust e quindi voglio che l’informazione rilasciata sia del minor costo possibile.

C certificati client, s = servizi/ certficati server

Informazioni del client:(policy)

c1← s1 vuol dire: ti do c1 se ti mi hai già dato s1 (se sono già in s1)

Server:

s1← c3∨ c4 per avere s1 devi darmi c3 o c4

La rivelazione delle credenziali è regolata da una policy che specifica i prerequisiti che devono

essere soddisfatti. Obiettivo: raggiungere una trusted negotiation tu mi dai una cosa in cambio di un’alltra.

Soluzione: Cammino minimo

Svantaggi:

Le informazioni fornite sono di granularità grossolana, non ho focus sulle preferenze del client.

Il cammino minimo comprende costi di server + client (combinazione non sempre utile)

**Point-based Trust Management Model**

Il server da dei punti alle credenziali del client, il client deve raggiungere una certa soglia per poter essere garantito di un determinato servizio.

I punti che il server associa alle credenziali sono privati e i punti che il client associa alle credenziali sono privati.

Nel senso che: il client deve cercare di soddisfare la soglia imposta dal server raggiungendo una sensibilità minima, dove la sensibilità è basata sui punti che il client dà alle proprie credenziali mentre la soglia da raggiungere è imposta dal server e i punti per raggiungerla vengono dati dal server rispetto alle credenziali del client (prende in considerazione il punteggio del server).

Il problema viene convertito in un knapsack problem(problema dello zaino) e risolto con tecniche di programmazione dinamica.

Svantaggi:

La soluzione può modellare solo le caratteristiche additive della privacy.

Il client e il server devono concordare sull’universo dei possibili tipi di credenziali (può essere

compressa la confidenzialità delle policy del server).

Supporta solo specifiche con granuralità grande. Associazioni sensibli così come i rilasci proibiti

non possono essere espressi.

Mette focus sulla negoziazione preferendola al controllo del client.

**Logic-based Minimal Credential Disclosure**

Il rilascio di credenziali è regolamentato da una policy data e si ha la semplificazione che ogni credenziale sia composta solo da un attributo.

Mettendo insieme le policy del rilascio di credenziali si possono creare dei path di negoziazione che mi permettano di arrivare ad ottenere il servizio richiesto. I path si creano mettendo insieme ciò che vuole il server. (cbdate ←cbbb significa che per dare via cbdate devo aver ricevuto cbbb)

In base al path migliore che si presenta sceglierò esso per arrivare ad ottenere il servizio richiesto.

Il path pi domina pj se i valori di pi sono tutti migliori uguali ai valori di pj e c’è un valore per cui pi è strettamente migliore di pj.

Non avere disclosure di una credenziale è considerato strettamente migliore di avere disclosure di quella credenziale (0 migliore 1) 0-> no disclosure 1-> disclosure

Le user preferences possono essere indicate da delle gerarchie.

Più si è in alto nella gerarchia più quella credenziale è preferita al rilascio rispetto a quelle più in basso nella gerarchia. (però detto così è brutto gli elementi della gerarchia sono legati da freccie -> se ho ci -> cj allora preferisco rilasciare prima ci che cj e ci starà più in alto nella gerarchia)

Dopo aver applicato concetto di dominanza e di user preferences ai path per la disclosure delle credenziali resteranno pochi paths tra cui l’utente potrà scegliere, la scelta su di essi dipende esclusivamente dall’utente, magari applicando criteri sulla sensibilità personale dei dati da rilasciare

Svantaggi:

Non si supportano credenziali che non voglio rilasciare,

Scegliere rispetto ad un gruppo di attributi non è sempre facile,

Le credenziali sono composte da un solo attributo.

**Privacy Preferences in Credential-based Interactions**

Si ha la possibilità per l’utente di gestire le proprie credenziali e fornirle in modo specificato al server. Le proprietà e credenziali sono organizzate nel portfolio dell’utente.

Credenziale: certificato firmato da una terza parte, certifica un’insieme di proprietà, ha un emittente, un id e un tipo.

Dichiarazione: proprietà mantenuta come credenziale self-signed

La gerarchia delle credenziali è di tipo: ISa.

Ie proprietà possono essere:

Indipendenti dalla credenziale: dipendono solo dall’owner, dati riguardanti l’owner

Dipendenti dalla credenziale: es: numero di carta di credito, non dipendono dall’owner ma dalla credenziale in sé

Le credenziali possono essere:

Atomiche: rilasciano tutto insieme, non posso separare il rilascio di proprietà

Non-atomiche: le proprietà possono essere rilasciate singolarmente, si può utilizzare la proof of possession (idemix - Uprove)

Una Disclosure cioè rilascio è un sottoinsieme di proprietà del portfolio dell’utente che sono:

Certificabili: ogni proprietà è certificata da una credenziale

Atomiche: se una proprietà legata ad una credenziale atomica viene rilasciata allora l’intera credenziale viene rilasciata(tt le proprietà)

La sensibilità di un portfolio cambia a seconda dell’utente, possiamo assumere di avere una sensibilità riguardante le credenziali e una riguardante le proprietà della credenziale, la sensibilità è trattata come un valore intero ed esse si possono infatti sommare tra loro.

La sensibilità può anche riguardare oltre che credenziali e proprietà anche una associazione tra proprietà o credenziali che siano. Quindi es: rilasciate due proprietà specifiche avrò una sensibilità aggiuntiva dovuta al rilascio di quelle due proprietà (può essere positiva o negativa)

In questo modello è possibile avere una forbidden view: due proprietà non possono essere rilasciate insieme.

Limitazione sul rilascio: al più n proprietà tra quelle evidenziate possono essere rilasciate. Nel rilascio finale si intende (disclosure)

Una disclosure è valida se nessun vincolo sul rilascio è negato: forbidden view e limitazione sul rilascio.

La sensibilità di una disclosure è la somma del valore delle etichette: (proprietà, credenziali, associazioni). I vincoli non hanno valori, se non sono rispettati annullano la disclosure.

Richiesta del server:

Il server vuole determinate proprietà/credenziali, la modellazione di una richiesta del server viene fatta tramite congiunzioni di termini, ad esempio: voglio che mi rilasci carta di credito e carta di identità. (A and B)

Il server mi darà un’insieme di richieste e il client avrà la possibilità di scegliere di rispondere con il rilascio più opportuno per il proprio portfolio.

Con rilascio più opportuno si intende avere il rilascio di valore minimo possibile, il valore è dato dalla somma delle etichette su proprietà credenziali e associazioni. Se non esiste nessun rilascio minore di quelle attuale allora esso è quello minimo possibile. (problema NP-hard)

Si risolve tipicamente avendo una max-SAT rapresentation e dandolo in pasto ad un SAT-solver.

Comunque questo approccio non considera release di dati successive tra loro.